Real-time reference trajectory adaptation for haptic shared control during teleoperation

MSc Thesis

S.A. Seuren

Delft University of Technology

TU Delft

iit ISTITUTO ITALIANO DI TECNOLOGIA

Cover image: Sarcos robot 'Guardian GT'. [1]

Real-time reference trajectory adaptation for haptic shared control during teleoperation

MSc Thesis

by

S.A. Seuren

to obtain the degree of

Master of Science
in Mechanical Engineering

at the Delft University of Technology,
to be defended publicly on Tuesday August 14, 2018 at 14:00.

Student number:	4083121
Project duration:	September 25, 2017 – August 14, 2018
Thesis Committee:	Prof. dr. ir. D. A. Abbink
	Dr. ir. H. Boessenkool
	Dr. ir. M. Kok
	Dr. ir. M. J. A. Zeestraten

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

This thesis contains a research that is the result of a collaboration between the Delft University of Technology and the Istituto Italiano di Tecnologia (IIT) in Genoa, Italy. A lot of knowledge and support on the technical side of the research was available at IIT, where the first three months of the research has been performed. The Delft University of Technology has a haptics lab that has a lot of knowledge available on the human-machine interaction. The last part of the research, including the experiment, has been performed at the Delft University of Technology.

This thesis is submitted for the Master of Science degree in Mechanical Engineering at the Delft University of Technology. The research and its results are presented in a scientific paper that is part of this thesis. The appendices contain more in-depth descriptions of the design choices and a guideline on using the experimental setup.

S.A. Seuren
Delft, August 2018

TU Delft Delft University of Technology

iit ISTITUTO ITALIANO DI TECNOLOGIA

Acknowledgements

After finishing the first year of my master Biomechanical Design, I was destined to do my master internship abroad. I spent a big amount of the summer of '16 trying to find a place where I could do this internship. Luckily I saw a post in the Biomechanical Design Facebook group in the beginning of August, stating that there was an internship vacancy in Tokyo, Japan. In order to apply for this internship, I needed a referral letter from Pieter Jonker. So I went to his office to discuss the internship application and little did I know that this would also be the meeting where my graduation research would start to take shape.

While Pieter was writing the referral letter we chatted about my search for an internship and my interests. When I mentioned Istituto Italiano di Tecnologia (IIT) in Genoa as one of the institutes that I was interested in, he said: *"Well, I can get you in contact with one of my alumni who is doing his PhD at IIT and maybe you can do your graduation research there."*. He asked me to send him an e-mail with my interests on the spot, which he forwarded to my (future) supervisor Martijn Zeestraten saying: *"And always end your e-mail with a smiley face to keep it informal :-)"*. This mail was the start of a process that eventually resulted in the master thesis in front of you.

I would like to thank Pieter for helping me with arranging my internship in Japan and kick-starting my graduation research. Several e-mails and Skype calls after the initial e-mail Martijn and Sylvain Calinon wrote a research proposal that also got David Abbink on board. David was willing to supervise the research from the haptics perspective, while Martijn was willing to do this from the technical perspective. I want to thank David and Martijn for all the valuable feedback throughout this research and I also would like to thank Sylvain and Martijn for taking the time to prepare the assignment that really fitted my interests and for arranging that I could partially conduct this research at IIT.

The experiment of my research was conducted on the 'Munin', an in-house built teleoperation device, that required a lot of work before it was suited for use in an experiment. This process had its ups and downs, but with the help of Henri Boessenkool it went relatively smooth. Since Henri did his own research on the same device, he could provide a lot of in-depth advice on how to implement certain parts of the model. I'm really thankful to Henri for the help during the setup of the experiment and the feedback on my thesis in the later stage of the research.

During my research I spent a lot of time in the Delft Haptics Lab. I would like to thank every member of the lab for the fun discussions, bouldering sessions, kip-knacks lunches and all the valuable feedback throughout my research.

Last but not least I would like to thank my family and girlfriend Michelle for all the support during the project. Spending time with them was a pleasant way to clear my mind once in a while, and allowed me to continue the research with a fresh mind.

S.A. Seuren
Delft, August 2018

Contents

1

Paper

Real-time reference trajectory adaptation for haptic shared control during teleoperation

Stijn A. Seuren, Henri Boessenkool, Martijn J. Zeestraten, and David A. Abbink, *Senior Member, IEEE*

Abstract—Haptic Shared Control (HSC) can improve operator performance during teleoperation. HSC uses assistive forces on the master device to guide the operator's control input towards a reference trajectory. However, the reference trajectory might be incorrect due to inaccuracies in the sensed or modeled environment and thus not match the intended trajectory of the operator, resulting in force conflicts that can potentially reduce performance. This paper describes the design and evaluation of a haptic shared controller whose reference trajectory adapts in real-time to resort force conflicts. Different adaptive controllers have been designed and compared in simulation. The final controller is built on a Task Parameterized Gaussian Mixture Model and adapts using an Incremental Gaussian Mixture Model. The controller is evaluated in an experiment ($n = 16$) where subjects performed a planar teleoperation task that involved moving the slave through a narrow corridor. The subjects were assisted by HSC with a correct reference (cHSC), HSC with a wrong reference (wHSC), HSC with a wrong reference that adapts in real-time (waHSC) and a manual condition. The waHSC condition was expected to result in lower control effort, compared to wHSC, and induce better performance compared to wHSC and the manual condition. Within-subjects results showed that waHSC decreases control effort compared to wHSC while the performance remained similar. Manual control had better performance than wHSC and waHSC, while cHSC outperformed all other conditions. Even though the conditions were tested in a balanced order, we observed a strong effect of the order of conditions: between-subject results showed improved performance, lower control effort and better subjective results for waHSC over wHSC when wHSC was executed first. Upon first execution of waHSC, improved performance and self-reported satisfaction were realized for wHSC over waHSC. The results show that, in the event of a force conflict, an adaptive controller is a promising alternative to temporarily adjusting the HSC stiffness.

Index Terms— Teleoperation, haptic shared control, conflict, reference adaption, Task Parameterized Gaussian Mixture Model, Incremental Gaussian Mixture Model

1 Introduction

Teleoperation devices consist of an operator interface, the *master*, and a manipulation robot, the *slave*, which are connected by a *controller*. These devices allow humans to remotely manipulate objects in environments that are not directly accessible. This can be due to safety, as applies for environments like space [1, 2], deep sea [3, 4] or nuclear sites [5], or for dimensional reasons like in minimal invasive surgery [6, 7, 8] or micro manipulation [9].

Achieving good teleoperation performance remains a challenge because of limited haptic- and visual feedback. Haptic Shared Control (HSC) can improve this performance. It assists the operator during the task execution by generating haptic force feedback that is based on a programmed reference in the controller. Application of HSC improves performance and reduces control effort during task execution [10, 11, 12].

However, these benefits are gained under the assumption that the reference trajectory of the controller is correct. Whenever the reference trajectory of the controller is incorrect, the controller will provide incorrect HSC to the operator of the system. The incorrect HSC will result in unnecessary conflicts between the operator and the controller. Liabilities in creating a correct reference trajectory can be the sensors that provide the controller with the necessary information, the processing of this information or a dynamically changing environment.

While incorrect HSC doesn't necessarily result in worse teleoperation performance [13], it does increase the control effort [14]. If the incorrect reference trajectory is consistent, calibration of the system may suffice as a remedy. However, if the source for incorrect reference varies in time, calibration might be too much effort. Conflicts that are caused by short-term incorrect reference trajectories can be reduced in two ways: by changing the Level of Haptic Authority (LoHA) or by changing the reference where the HSC is based upon.

The LoHA determines to which degree the operator is guided by the controller [15, 16, 17]. As stated by Abbink et al. [17]: "...for high stiffness—a high LoHA—the performance is expected to be improved —of course, only as long as the human operator agrees with the automation and there are no automation failures. For low stiffness—a low LoHA—there is not much support from the system, so little performance increase is expected, but the system will be easy to overrule.". Reducing the LoHA could therefore be an option to decrease the conflicts that arise when the automation fails and thus generates incorrect reference trajectories. But, changing the LoHA can be considered symptom management and implies accepting the incorrect reference trajectory.

Another option is to adapt the reference trajectory on which the HSC is based. This could reduce conflicts while maintaining the benefits of HSC. De Jonge et al. [18] showed that a trial-by-trial adaption of a reference trajectory can decrease the control effort opposed to a static reference. This

- *S.A. Seuren, H. Boessenkool and D.A. Abbink are with the Department of Cognitive Robotics, Faculty 3mE, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands.*
- *M.J. Zeestraten was with the Department of Advanced Robotics, Istituto Italiano di Tecnologia, Via Morego 30, 16163 Genova, Italy.*

trial-by-trial adaption is considered offline adaption; the reference trajectory is adapted for use in a future execution of the same trial and not for use in the current trial. For reducing short-term conflicts the adaption should occur online. No literature was found related to controllers that are able to adapt the reference trajectory online.

The goal of this paper is to design a haptic shared controller that, by adapting the reference trajectory online, improves the task performance and reduces the control effort in the event of a short-term incorrect reference trajectory. This controller should decrease the number of conflicts between the operator and the controller while maintaining the benefits of HSC. The controller will be evaluated experimentally, for which is hypothesized that an incorrect but adapting reference will result in better performance (H1) and lower control effort (H2) compared to an incorrect and non-adapting reference. Finally, it is hypothesized that the performance with incorrect but adapting HSC is better than manual teleoperation (H3).

The paper starts with Section 2 that discusses which method is chosen for generating the reference trajectories and how these trajectories will be adapted. Section 3 starts with describing the experimental setup. Subsections 3.2 to 3.6 discuss how the chosen methods are modified, implemented and tuned in the controller for use in the experimental setup. The retrieved experimental data are analyzed in Section 4. The notion of an adaptive controller and the designed controller will be discussed in Section 5. The conclusions are described in Section 6.

2 Design of an Adaptive Controller

The proposed adaptive controller can be divided into two parts: one that generates a reference trajectory and one that can adapt this reference trajectory. The parts have to work alongside each other, but have different criteria.

2.1 Generating a reference trajectory

The reference trajectory is adapted by the adaptive controller in every update. As a result, the magnitude or direction of the guidance force will also change with every update. To make the adaptation appear smooth, it should be executed at a high frequency [19].

Furthermore, the controller should be prone to adaptation. If conflicts arise, the control parameters should allow being changed in such a way that the reference trajectory makes the desired adaptation.

The controller in this research uses a Task Parameterized Gaussian Mixture Model (TP-GMM) [20] for the generation of the reference trajectories. TP-GMM is a probabilistic model which originates from the field of imitation learning. This method is model-based, meaning that it does only require several parameters to function, and is therefore capable of quickly generating the reference trajectories. Furthermore, the process from demonstration data to reference trajectory allows for adaptation on several levels, as seen in Fig. 1. Finally, TP-GMM has already been applied in several HSC systems [21, 22, 23]. For these reasons, TP-GMM is a suitable method for generating reference trajectories in an adaptive controller.

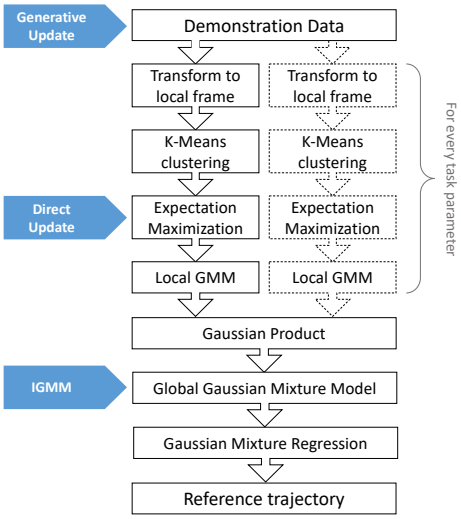


Fig. 1. Schematic overview of the Task Parameterized Gaussian Mixture Model (TP-GMM). The locations where Generative Update, Direct Update and the Incremental Gaussian Mixture Model (IGMM) can adapt an existing model are marked with the blue pointers.

The TP-GMM is trained with demonstration data that consist of multiple recordings of the movements that require guidance. Each data point contains the location of the slave (x, y) and a timestamp t. During training, the demonstration data are clustered (K-Means) and transformed to k Gaussian distributions (Expectation Maximization) relative to each of the local frames defined by the task parameters. A task parameter defines the local reference frame in the global reference frame with a position vector $\boldsymbol{b}$ and rotation matrix $\boldsymbol{A}$. The k Gaussian distributions, or 'states', in the local frame p are defined by a state mean $\boldsymbol{\mu}_{k,p}$ and covariance $\boldsymbol{\Sigma}_{k,p}$. The local states combined form a local GMM. By using the task parameters, the local GMMs can be combined into a global GMM ($\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$) using the Gaussian product. This global GMM serves as input for Gaussian Mixture Regression (GMR) [24]. For every timestamp t the GMR algorithm finds the most likely corresponding (x, y) position, which in sequence form the reference trajectory. Fig. 1 shows a systematic overview of this process.

2.2 Adapting a reference trajectory

Using TP-GMM, reference trajectories can be adapted on different levels. Three methods for adapting a GMM have been found in literature. Calinon and Billard [25] describe two methods for adapting an existing GMM: Generative Update and Direct Update. The third method is introduced in the work by Engel and Heinen [26] and adapts a GMM using an Incremental Gaussian Mixture Model.

The Generative Update samples new demonstration data from an existing reference trajectory. A new GMM will be constructed based on this sampled data combined with the current movement data. GMR can then be used to create an adapted reference trajectory.

Direct Update uses a modified Expectation Maximization algorithm to weigh the summed likelihood of the original demonstration data versus the summed likelihood for all the data points of the slave movement. Depending

on the relative weights, Direct Update will create a new GMM with which the adapted reference trajectory can be generated through GMR.

The third method for adapting a GMM is described by Engel and Heinen [26]. They introduced the Incremental Gaussian Mixture Model (IGMM), which "creates and continually adjusts a Gaussian Mixture Model consistent to all sequentially presented data.". IGMM checks the likelihood that the current slave location belongs to state k and adjusts the locations of the state means and the covariance matrices accordingly.

The main difference between the three methods is the location within the reference trajectory generation where the methods make changes (Fig. 1). Based on these different levels it can be hypothesized that IGMM will be the least computationally expensive for every update, followed by Direct Update and Generative Update respectively.

2.2.1 Simulation of available adaptation methods

Each of the methods is tested in a simulation environment to test this hypothesis and to get insight into the behavior of each method. The simulation was conducted in MATLAB and is based on code provided by Calinon [20]. The training of the TP-GMM was performed using a cursor in the MATLAB environment depicted in Fig. 2a. Three task parameters –start, obstacle and end– were used for the training, of which the starting point and endpoint had 3 variations. A total of 9 demonstrations were performed to cover all the possible paths. The collected demonstration data were re-sampled in order for all the demonstrations to have the same length of 100 data points. Finally, the TP-GMM was trained with the re-sampled data.

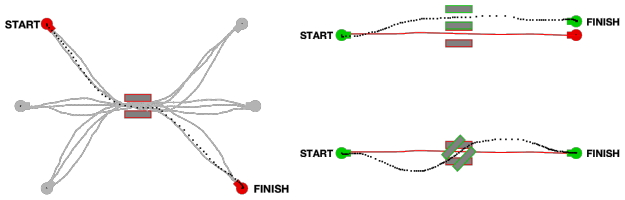


Fig. 2. The GMM training environment (a) and the simulation environments with incorrect reference trajectory and the slave movement (b)

Using GMR, a reference trajectory for a predetermined set of task parameters was generated based on the trained TP-GMM. The resulting reference trajectory is considered the incorrect reference trajectory and is shown in red in Fig. 2b. The correct environment is simulated by changing some task parameters. Two correct environments were tested for determining the behavior of the adaptation methods: an environment where the obstacle and end point were translated and an environment where the obstacle was rotated. These correct environments are shown in green in Fig. 2b. The slave movement is performed in the correct environment with the cursor in two different ways: one movement at a constant speed and one movement with varying movement speeds. These slave movements are also logged and are re-sampled to 100 data points.

The relative computational cost of each method has also been tested using simulation. For this simulation, the user movement data were re-sampled to different sizes ranging from 50 to 500 points. Each of these data sets was used 10 times as input for the three adaptation methods. The required time to finish the adaptation was logged for each simulation.

2.2.2 Simulation results

The logged times for each data set are averaged and a variance is calculated over the repeated measures. The result of the simulation test on relative computational costs is visualized in Fig. 3. The hypothesis that Generative Update would relatively be the most expensive, followed by respectively Direct Update and IGMM, is confirmed by these results.

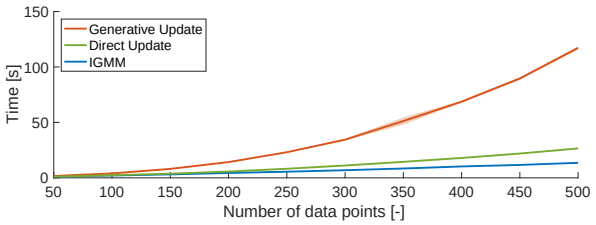


Fig. 3. The computation time per adaptation method for increasing data points in the user movement. The variance is displayed by with a shade around the line.

The general features of each method are assessed and their behavior is visually observed using the simulation environments of Fig. 2b.

Generative Update showed to be unusable for application in an adaptive controller for several reasons. The main reason is the shock wise adaptation of the reference trajectory. Since a new GMM is constructed for every update, the generated reference trajectories vary slightly for every update. If this method would be implemented in HSC it would result in a vibrating feedback force and is therefore not suitable for this application. Moreover, Generative Update is computationally the most expensive relative to the other methods and becomes more expensive with the growing amount of slave movement data. Adapting the reference trajectory multiple times per second might be unfeasible with the hardware available. Finally, sampling new data from a reference trajectory doesn't represent the original model unless a lot of samples are generated, which in turn adds to the computation time when constructing a GMM. As a result, information from the original GMM will get lost when Generative Update is used. In terms of behavior, Generative Update showed unable to handle varying movement speeds, but was able to adapt the reference trajectory to the rotated and translated obstacle.

Direct Update showed promising results in the simulation. This method is less computationally expensive than Generative Update, but still requires all the slave movement data and therefore also becomes more computationally expensive with time. In contrast to Generative Update, Direct Update is not able to cope with the rotated obstacle but was able to handle varying movement speeds.

The most suitable for implementation in HSC is IGMM and will therefore be used in this research. This method only

requires the current slave position, not the past movement, to adapt the reference trajectory. As a result, the speed of the adaptation won't change with time or movement distance. Moreover, this method requires the least computation power per update relative to the other methods. Similar to Direct Update, IGMM can't cope with the rotated obstacle, but can handle varying movement speeds. The limits of IGMM will be elaborated upon in Section 5.3.

3 METHOD

3.1 Experimental setup

The setup used is an in-house built planar teleoperation device called the 'Munin' [27]. This device can provide haptic feedback up to 5N/1Nm (x, y, θ), has an identified slave inertia of 260 ± 10g and has 3 degrees of freedom of which the rotation is disabled for calibration purposes. The master and slave are linked by a position-error controller $(K_p : 400, K_d : 0.02)$ running on a 1kHz real-time computer. A webcam is placed above the slave-side of the system to provide visual feedback to the operator.

Fig. 4. The master (L) and slave (R) of the 'Munin' telemanipulator.

3.1.1 Task boundaries

The operator has to move the slave side between the boundaries shown in Fig. 5. These boundaries are printed on an A3 underlay to provide a visual reference. The width of the diagonal sections is 1cm, the horizontal section has a width of 1.5cm. The location of the boundary is calibrated into the model by entering the slave's XY coordinates at the points that define the boundary.

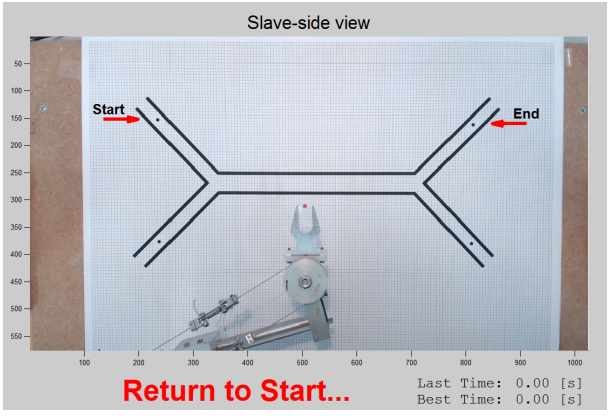


Fig. 5. The webcam video feed

3.2 The adapting haptic shared controller

The controller will use TP-GMM to construct a model of the environment, IGMM to adapt the model online and GMR to generate reference trajectories from the (adapted) model. The following subsections will elaborate on how each step is implemented and what parameters are used.

3.2.1 Task Parameterized Gaussian Mixture Model

The TP-GMM is trained prior to the experiment in order to guarantee a correctly trained model during the experiments. The amount of states in the GMM is set to $k = 3$ and the task parameters are defined by the location and orientation of the start, finish and middle of the path. The middle of the path is fixed in all situations, but the starting position and end position can vary resulting in 4 possible paths (Fig. 6). A tunable regularization term R is added to the diagonal of each covariance matrix $\mathbf{\Sigma}_{k,p}$ prior to the Gaussian product. A higher value for R increases the variance of the states in the GMM, resulting in a less accurate but smoother reference trajectory from the GMR.

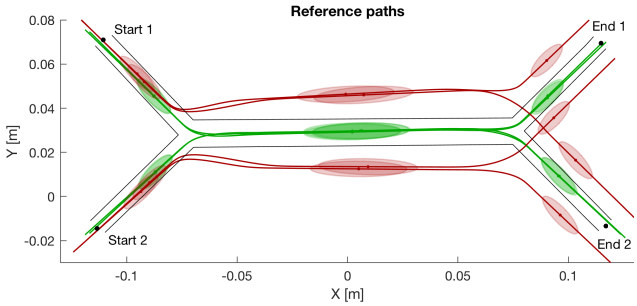


Fig. 6. Correct and incorrect reference paths

3.2.2 Incremental Gaussian Mixture Model

The used method to adapt the GMM is a modified version of the Incremental Gaussian Mixture Model (IGMM). The IGMM method was not designed to be implemented in HSC and requires modifications before it can be used to adapt a reference trajectory. The original IGMM introduced a new state whenever the likelihood that a new data point belonged to one of the existing states was below a defined *'novelty threshold'*. The modified IGMM doesn't include such threshold and is not capable of introducing new states to the global GMM. Further modifications include addition of lines 4, 7, 9, 13 (see Algorithm 1) and the changed value of $sp_k(0)$.

The IGMM algorithm consists of a loop that adapts each state separately. The loop starts with estimating the likelihood $p(k|\boldsymbol{x})$ that the current slave location $\boldsymbol{x}$ belongs to state k. This likelihood is added to the saturation parameter sp_k^* in line 3, which determines how eager a state is to adapt. The initial value for the saturation parameter $sp_k(0)$, or presaturation, can be tuned to prevent a state from adapting too early. Contrarily, line 4 limits the saturation parameter to the maximum value sp_{max} to prevent over-saturation of the state which can limit its adaptation. In line 5 the likelihood is divided by the saturation parameter to retrieve ω_k^*, which reflects the confidence in the current slave location: a higher

Algorithm 1 Modified IGMM adaptation algorithm

1: **procedure** IGMM ADAPTATION($x, sp, \boldsymbol{\mu}, \boldsymbol{\Sigma}$)
2: **for** $k = [1...k_{total}]$ **do**
3: $sp_k^*(t) = sp_k(t-1) + p(k|\boldsymbol{x})$
4: $sp_k(t) = \min(sp_k^*, sp_{max})$
5: $\omega_k^* = \frac{p(k|\boldsymbol{x})}{sp_k}$
6: $\omega_k = \alpha\omega_k^*$
7: $\boldsymbol{e}_k = \boldsymbol{x} - \boldsymbol{\mu}_k(t-1)$
8: $\Delta\boldsymbol{\mu}_k^* = \omega_k \boldsymbol{e}_k$
9: $\Delta\boldsymbol{\mu}_k = \min(\Delta\boldsymbol{\mu}_k^*, \Delta\boldsymbol{\mu}_{max})$
10: $\boldsymbol{\mu}_k(t) = \boldsymbol{\mu}_k(t-1) + \Delta\boldsymbol{\mu}_k$
11: $\boldsymbol{e}_k^* = \boldsymbol{x} - \boldsymbol{\mu}_k(t)$
12: $\boldsymbol{\Sigma}_k^*(t) = (1-\omega_k)\boldsymbol{\Sigma}_k(t-1) + \omega_k \boldsymbol{e}_j^* \boldsymbol{e}_j^{*T} - \Delta\boldsymbol{\mu}_k \Delta\boldsymbol{\mu}_k^T$
13: $\boldsymbol{\Sigma}_k(t) = \text{AffineRotation}(\boldsymbol{\Sigma}_k^*(t), \boldsymbol{\Sigma}_k(t-1))$
14: **end for**
15: **return** $sp, \boldsymbol{\mu}, \boldsymbol{\Sigma}$
16: **end procedure**

value for ω_k^* results in more adaptation of the related state towards the slave's location. Line 6 introduces the adaptation coefficient α that is a gain for ω_k^* and allows for tuning of the overall adaptation rate. The required translation of the state mean can then be calculated in line 8 by multiplying ω_k with the error $\boldsymbol{e}_k$ between the slave and the respective state mean. The actual adaptation of the state mean in line 10 is limited in line 9 by the maximum displacement per update, which is defined by $\Delta\boldsymbol{\mu}_{max}$. Lines 11 and 12 are derived from the Gauss-Wishart distribution and calculate the updated covariance matrix $\boldsymbol{\Sigma}_k^*(t)$.

The adaptation of $\boldsymbol{\Sigma}_k(t-1)$ to $\boldsymbol{\Sigma}_k(t)$ is limited to rotation using an algorithm we call 'AffineRotation'. AffineRotation calculates the orientation difference between the two covariance matrices using the biggest eigenvector of each matrix. This difference is used to create a rotation matrix with which $\boldsymbol{\Sigma}_k(t-1)$ is rotated, resulting in $\boldsymbol{\Sigma}_k(t)$. The adaptation limited to rotation because a varying size can lead to stretching of the states while trailing the slave. Moreover, the covariance matrices represent a 'confidence interval' based on the original training so varying their size would discard this information.

3.2.3 Gaussian Mixture Regression

The demonstration data set that was used to train the TP-GMM consisted of multiple recorded movements between the start and end positions. Each movement consisted of 1000 data points, where the time labels of the demonstration data were linearly spaced between 1 and 1000. The input to the GMR therefore consists of 1100 points between $-50 \leq t \leq 1050$ in order to extend the reference trajectory beyond the start and end positions.

3.2.4 Generation of guidance forces

The reference trajectory that is created by the TP-GMM and IGMM is used as the reference for the HSC. Most commonly this is done by applying a stiffness K_{hsc} on the error between the slave's location and the closest point on the reference trajectory. For this research, we employed an HSC design from previous work in our group on this manipulator [18, 10, 28]. In this HSC design the guidance is based on the future location of the slave by using a look-ahead controller. This predictive guidance towards a reference trajectory has shown to improve performance compared to regular force fields [29]. Equation 1 shows how the future location of the slave is calculated, whereas Equation 2 shows the calculation of the HSC forces. In this equation is x_{ref} the point on the reference trajectory that is nearest to the future location of the slave.

$$\boldsymbol{x}_{slave}(t+t_{la}) = \boldsymbol{x}_{slave}(t) + t_{la}\boldsymbol{v}_{master}(t) \quad (1)$$

$$\boldsymbol{F}_{hsc}(t) = K_{hsc}(\boldsymbol{x}_{ref} - \boldsymbol{x}_{slave}(t+t_{la})) \quad (2)$$

The tuning of the look-ahead time t_{la} and HSC stiffness K_{hsc} depend on the environment and type of task. The task in this experiment is to follow a trajectory while refraining from hitting the boundaries around the trajectory. As the boundaries leave little movement space (Fig. 5), a small error requires more guidance force to prevent the slave from hitting the boundary. A high HSC stiffness is therefore required, which is set at 200 N/m. Moreover, since small errors result in a high guidance force, the look-ahead time should be relatively low. The look-ahead time is tuned by hand and set to 0.2 seconds. Lastly, the controller is programmed in such a way that no data points of the reference trajectory can be skipped. The haptic shared controller has to go through every data point in the reference trajectory, which prevents sudden changes in the direction of the guidance.

3.2.5 Tuning

The tuning of the controller is different for each environment, demonstration set and refresh rate of the system. The (train) environment and demonstration data have a big influence on the covariance matrices in the GMM and thus the resulting adaptation. The update frequency of the system has a linear relation with the adaptation rate of the reference trajectory. The setup used runs on a 1kHz computer, so the controller is tuned accordingly.

Several things were kept in mind during tuning of the IGMM. First of all, adaptation should not result in a wrinkle in the reference trajectory for different movement speeds or strategies (R1). Wrinkles result in unpredictable guidance forces and therefore should be avoided. The second requirement is a high adaptation rate (R2). The trials in the experiment are relatively short in terms of time and distance, therefore quick adaptation is required to emphasize the effects of the adaptation.

Table 1 shows the parameters that are available for tuning. The regularization term R is chosen as high as possible as long as the reference trajectory of the cHSC condition stays within the boundaries and doesn't cut the corners. A higher R results in smoother reference trajectories that are less likely to result in kinks (R1) and it makes states more susceptible to adapt (R2) due to the higher $p(k|\boldsymbol{x})$ value. The adaptation coefficient α is set at 0.3, which is relatively high (R2). The realized adaptation is limited using parameters $sp_k(0)$ end $\Delta\mu_{max}$ (R1), which are manually tuned and set to 120 and $1e-5$ respectively. These parameters are tuned manually and respectively prevent a state from adapting too early and limit the absolute displacement of state means

TABLE 1
Used parameter values

General parameters

Variable	Value		Description
K_{hsc}	200	[N/m]	HSC stiffness
t_{la}	0.2	[s]	Look-ahead time
R	1e-5	[-]	Regularization term

IGMM parameters

Variable	Value		Description
α	0.3	[-]	adaptation coefficient
$sp_k(0)$	120	[-]	Presaturation of states
sp_{max}	2000	[-]	Max saturation of states
$\Delta\mu_{max}$	1e-5	[m]	Max displacement of state mean per update

per update, which serves R1. Finally, sp_{max} is also tuned by hand and is set to 2000. It is tuned in such a way that the middle state (Fig. 6) remains adaptable throughout the trial. Initially the middle state will move to the left, but if the value for sp_{max} is too high the state will not be able to move to the right due to the over-saturation.

3.3 Experiment protocol

The experiment consists of 1 training set and 4 sets with different conditions. Each experiment starts with a training set to familiarize the participants with the controls and the setup. The training set has no HSC so the participant won't be conditioned to one of the conditions. After the training set the participant will perform 12 trials in each of the 4 conditions, each with a controller named after their ability to adapt (a) and whether the initial reference trajectory was correct (c) or wrong (w).

1) **Manual**
   In this condition no HSC is present. As a result the operator only feels the inertia forces of the system.
2) **cHSC**
   The second condition uses correct HSC that does not adapt. The green lines in Fig. 6 show the correct reference trajectories.
3) **wHSC**
   The third condition uses a controller that also doesn't adapt, but has incorrect reference trajectories and thus pulls the slave to the wrong direction. The wrong reference trajectory is generated by translating the task parameters perpendicular to the path. The task parameters related to the starting point were left untouched to allow for a normal start. The red lines in Fig. 6 depict the translated reference trajectories.
4) **waHSC**
   The last test condition initially has the incorrect reference trajectories in every trial, just as the wHSC condition. However, this condition adapts online to the current movement using IGMM.

The different conditions were presented to the subjects in a balanced order using a Latin square. Each condition is followed by a digital 'Van der Laan' questionnaire [30] to collect subjective data on the usefulness and satisfaction of the condition.

Each trial consists of a movement from one of the two starting points on the left towards one of the two endpoints on the right (Fig. 5), making 4 possible paths. The order of the paths is randomized and each path is followed 3 times in each trial. The participant moves the slave between trials to the next starting position indicated by the red arrow. If the slave is on the starting position, the operator will feel vibrations in the master input device accompanied by a '*Start when ready*'-text on the webcam feed. The trial will then start automatically whenever the slave deviates more than 0.2 cm from the start and finish when the slave is within a 0.2 cm range of the endpoint.

The slave has to stay clear of the boundaries during the trial. Whenever a collision is detected during a trial, the operator will be notified onscreen and has to restart the trial. Only correctly completed trials add up to the 12 trials.

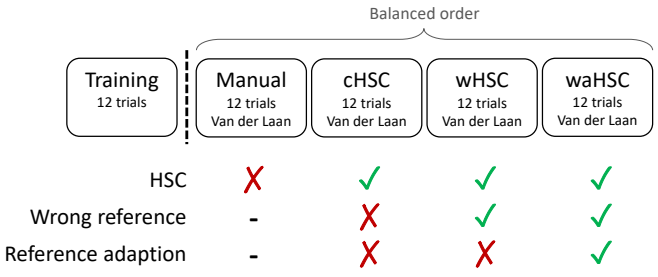


Fig. 7. Experiment overview

3.3.1 *Task instructions*

The participants in the experiment received written instructions prior to the experiment. They were instructed to *"Complete the trial as fast as possible without hitting the boundaries"*. Furthermore, the participants were asked to sit on a chair that was height adjustable to make sure their lower arm was in a horizontal position.

3.4 Measured variables and Metrics

The variables are logged at a sampling rate of 1kHz during the experiment. These data include position, velocity and force and also contain the HSC related data like the reference trajectory, state means, and covariance matrices. This data set forms the basis for the calculation of several metrics. The following metrics are used to measure the task performance:

- **Task completion time [s]**
  The time between leaving the starting position and reaching the end position.
- **Number of wall hits**
  The number of attempts that led to wall hits prior to completing a trial.

The control effort is quantified with the following metrics:

- **Y-reversals**
  The number of zero-crossings of the differentiated Y value. A higher number of reversals implies more

control effort, since more effort is put into correcting the position of the slave.

- **Maximum conflict force [N]**
  The maximum HSC interaction force between the operator and the controller within one trial.

Apart from the measured variables, subjective data is collected by means of a Van der Laan questionnaire.

3.5 Subjects

A total of 16 subjects participated in the experiment of which 75% were male. The average age was 24.8 ($\pm$2.3) years and they all were right-handed. Three subjects used the teleoperation setup before, the other subjects had no prior experience with this setup. The study was approved by the Human Research Ethics Committee of the Delft University of Technology. All the subjects gave their informed consent for the use of their data.

3.6 Analysis

3.6.1 Data processing

The relevant variables were logged at a frequency of 1kHz during the execution of the experiment. These raw data are pre-processed prior to the calculation of the metrics. The pre-processing included filtering all the location (x, y) and force data (F_x, F_y) using a 10Hz 4th-order Butterworth filter to remove the measurement noise. Also, all the wall hits in a radius of 1cm around the starting points were removed since they were often caused by accidental starts.

3.6.2 Statistical analysis

A two way ANOVA was used to determine the significance of the task completion time, Y-reversals and maximum conflict force metrics ($p < 0.05$). The used factors are the condition and the paths. For the wall hits the non-parametric Friedman test was used, since the wall hits weren't normally distributed as the hits can't get below 0. A post-hoc analysis is performed using the Bonferroni correction. The results of the Van der Laan questionnaire are analyzed using paired t-tests ($p < 0.05$).

4 Results

4.1 Raw data

Each subject performed 3 trials on each of the 4 possible paths for every experimental condition. Fig. 9 shows the trajectories of all the subjects, with four combinations of an experiment condition and path. The results of the 12 trials will be averaged for the metric calculation, assuming there are no significant differences in the results between each path. Contrarily to this assumption, there were significant differences between the paths.

These differences are the clearest in the raw data of the operator's movement speed. For the paths 2 and 3 there is a spike in the total speed between $0.05 < x < 0.1$, while this spike is not present in the raw data for paths 1 and 4 (Fig. 8). This spike in the total speed causes a slightly shorter task completion time, as seen in Table 2. The difference between the paths 2 and 3 and the paths 1 and 4 is the orientation of the reference trajectory relative to the user movement. In the paths 2 and 3, the operator takes the inner turn relative to the reference trajectory, while in paths 1 and 4 the operator takes the outer turn relative to the reference trajectory. This effect will be elaborated upon in the discussion.

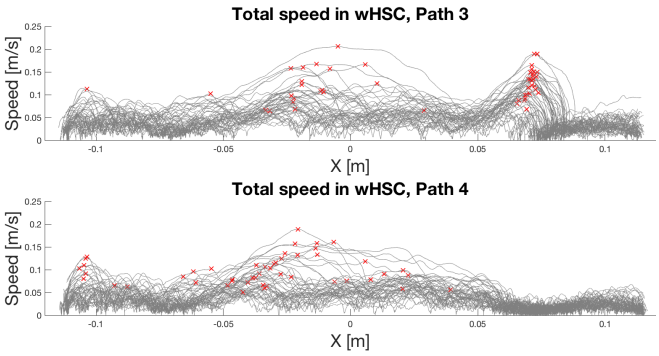


Fig. 8. Raw data plots of the total speed over the x location for paths 3 and 4. The red crosses denote the maximum speed within a trial.

The question remains whether the differences are grounds for excluding certain paths in the final results. The influence of this effect on the experiment is checked by excluding all the trials performed on path 2 and 3. This showed that the effect was relatively small and that the between condition results remained the same relative to each other. Also, this effect is considered an inherent property of wrong reference trajectories, so path 2 and 3 have not been excluded from the final results.

Another thing that has been analyzed in the raw data is how the reference trajectory adapted in waHSC. The trajectory adapted as expected in all the completed trials, meaning that the adapted reference trajectory is closer to the correct trajectory than it initially was (Fig. 9). However, in some trials where the slave hit the wall, the cause was likely that the adaptation didn't result in a smooth trajectory. The reason for this adaptation error will be discussed in Section 5.3.

Finally, the trajectory plots in Fig. 9 show that many hits in wHSC and waHSC occurred near the parting of the paths towards the end, while little hits occurred in the horizontal part of the paths.

4.2 H1: Performance wHSC vs. waHSC

We hypothesized that, when HSC was based on a translated initial reference trajectory, the designed adaptive controller would provide increased performance over a non-adaptive controller (H1). Contrary to our hypothesis H1, waHSC showed no benefits compared to wHSC: there was no significant difference in terms of performance or wall hits, as shown in 2 and as illustrated in Fig. 11.

The experiment was not designed to explore between-subject differences, but doing so resulted in interesting differences between wHSC and waHSC. The between-subject comparison shows that the order of waHSC and wHSC is relevant for both the measured and subjective results. With the assumption that the manual and cHSC trials had no significant effect on the task execution in the wHSC and waHSC conditions, a between-subjects one-way ANOVA

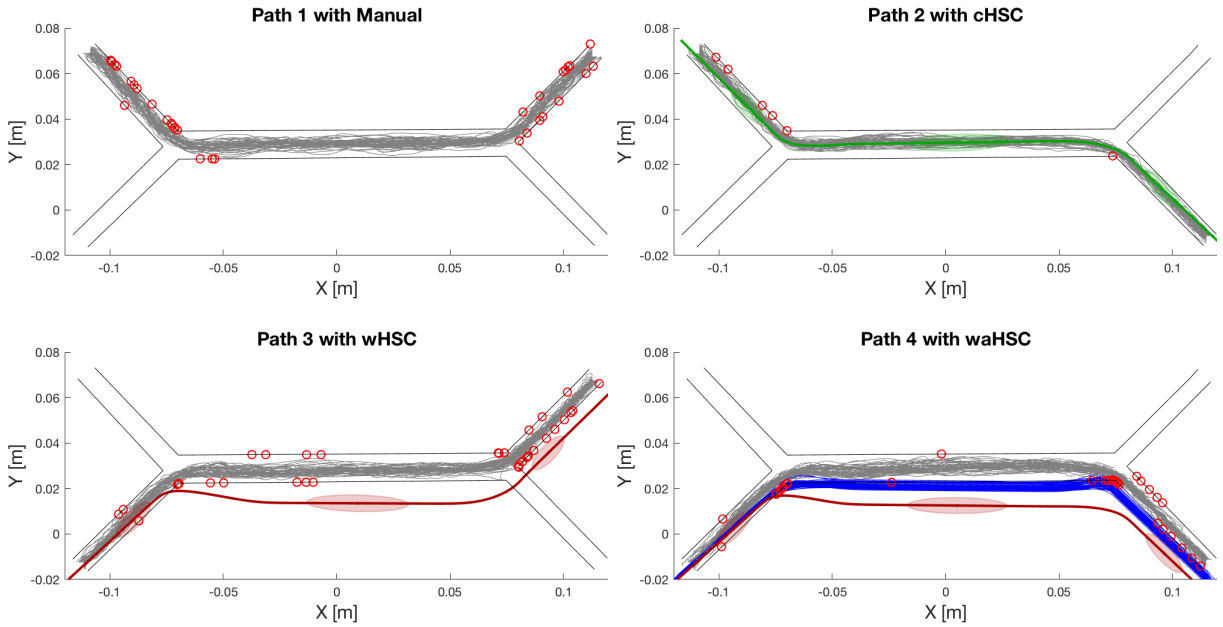


Fig. 9. Selection of raw trajectory plots. Each of the 4 conditions is plotted on one of the possible paths. A red 'o' represents the hit location of a failed trial.

test is conducted to quantify the influence of the different orders of condition execution.

When a subject ($n = 9$) first experienced wHSC followed by waHSC, the task completion time decreased significantly and waHSC was perceived as being significantly more useful ($p = 0.002$) and significantly more satisfying ($p = 0.015$). The other way around, when subjects ($n = 7$) performed the trials first in waHSC followed by wHSC, there was a significant decrease in task completion time and wHSC was perceived as more satisfying ($p = 0.011$), but no significant difference was found in the self-reported usefulness. The relevant graphs can be found in Fig. 10.

4.3 H2: Control effort wHSC vs. waHSC

We hypothesized that when there is an offset in the reference trajectory, the adaptive controller would result in a lower control effort compared to a non-adaptive controller (H2), where subjects had to continuously overcome the force guiding them towards the translated reference. The results confirm this hypothesis: waHSC significantly reduces the control effort compared to wHSC.

The control effort is measured with the metrics maximum conflict force and Y-reversals. For the maximum conflict force there is a significant difference between each of the conditions, with waHSC requiring less control effort than wHSC followed by cHSC. For the Y-reversals significant differences have been found between almost all the conditions, only manual and waHSC showed no significant difference. The wHSC condition had significantly the highest amount of Y-reversals, while cHSC had significantly the lowest amount of Y-reversals.

The between-subject results based on order of execution showed no different results for the maximum conflict force metric. For Y-reversals however, a significant difference was found when wHSC was performed before waHSC. No significant difference was found when wHSC was executed after waHSC.

4.4 H3: Performance Manual vs. waHSC

The third hypothesis stated that the adaptive controller with a wrong initial reference would have improved performance over the manual condition. However, the task completion time and wall hits show significant differences between manual and waHSC in favor of the manual condition. This means that whenever the initial reference trajectory is incorrect, it is better for the performance to disable the adaptive controller and use manual control to execute the task. H3 therefore can't be accepted.

5 Discussion

The results show that the adaptive controller is able to reduce the force conflicts that are caused by an incorrect initial reference trajectory. Opposed to our hypotheses, the adaptive controller was not able to increase the performance relative to the manual or wHSC condition. These results are comparable with the results in the work by De Jonge et al. [18]. Their trial-by-trial adapting controller also reduced conflicts, but didn't improve the performance relative to a non-adaptive controller.

The results also show some remarkable results, such as the differences between the paths and the differences between the orders of execution of waHSC and wHSC. These differences and other general topics on adaptive controllers will be discussed in the following sections.

5.1 Adaptation rate

The trials in the experiment were relatively short. It took operators roughly 10 seconds to complete a trial that spanned

TABLE 2
Table of the raw metric data per condition and per path, averaged over all the subjects. The number between the parentheses shows the standard deviation. The cells are shaded according to their relative values.

	Task completion time [s]				Number of wall hits [-]				Y-reversals [-]				Max. conflict force [N]		
	Man	cHSC	wHSC	waHSC	Man	cHSC	wHSC	waHSC	Man	cHSC	wHSC	waHSC	cHSC	wHSC	waHSC
P1	7.09	5.53	9.41	8.87	0.69	0.10	1.27	1.04	3.71	2.40	5.48	4.75	0.79	3.74	2.70
	(2.24)	(1.81)	(2.11)	(2.32)	(0.86)	(0.16)	(1.11)	(0.89)	(1.72)	(1.10)	(1.61)	(1.49)	(0.14)	(0.21)	(0.20)
P2	7.68	5.84	8.58	8.10	0.83	0.12	1.79	1.46	4.19	2.92	5.58	4.38	0.85	3.50	2.39
	(2.29)	(1.66)	(2.02)	(1.64)	(0.70)	(0.30)	(1.15)	(1.17)	(1.25)	(0.76)	(1.54)	(1.33)	(0.11)	(0.19)	(0.20)
P3	6.41	5.40	7.32	7.23	0.65	0.08	0.77	1.56	3.94	2.54	4.98	4.85	0.60	3.25	2.50
	(1.69)	(1.74)	(1.82)	(2.06)	(0.65)	(0.15)	(0.88)	(1.12)	(1.66)	(1.07)	(1.37)	(1.79)	(0.11)	(0.19)	(0.28)
P4	7.26	5.61	9.56	9.10	0.40	0.10	0.63	0.81	4.48	3.29	6.08	4.90	0.65	3.69	2.78
	(2.00)	(1.76)	(2.38)	(2.24)	(0.60)	(0.23)	(0.95)	(1.95)	(1.35)	(1.58)	(2.32)	(1.13)	(0.14)	(0.14)	(0.32)
All	7.11	5.60	8.72	8.32	0.64	0.10	1.11	1.22	4.08	2.79	5.53	4.72	0.72	3.55	2.59
	(2.07)	(1.71)	(2.23)	(2.16)	(0.71)	(0.21)	(1.11)	(1.35)	(1.50)	(1.19)	(1.75)	(1.43)	(0.16)	(0.26)	(0.30)

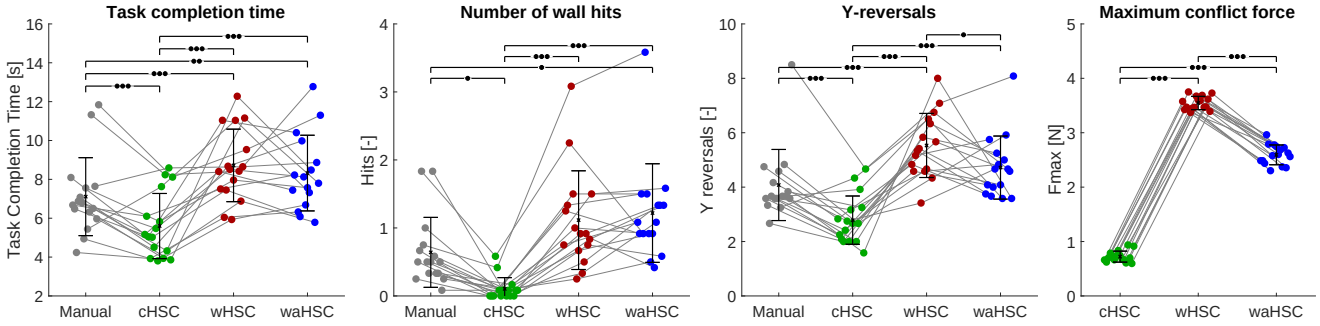


Fig. 11. Results for the different metrics. The connected dots represent one subject and the significance is denoted by bullets $(\bullet), (\bullet\bullet), (\bullet\bullet\bullet)$ that respectively represent p-values with $p \leq 0.05, p \leq 0.01, p \leq 0.001$ significance.

a distance of around 25 centimeters. Given this short trial, the adaptation rate was required to be relatively high in order to have the reference trajectory change enough to make a difference. The high adaptation rate had several implications on the behavior of both the operator and the controller.

For the operator it is hard to understand the intentions of the system since the system changes continuously. This could also be the reason for the different between-subject results for wHSC and waHSC: it might be harder to learn a control strategy with a continuously changing system than with a system that behaves consistently.

For the controller, a high adaptation rate results in tumultuous adaptation. The resulting bigger changes in the reference trajectory increase –in this adaptation method– the possibility that the reference trajectory wrinkles or adapts to undesired forms. Most of the controller tuning in this experiment was aimed at preventing wrinkling in the reference trajectory. Even when is assumed that the adaptation is stable, there are still other aspects to consider. Since the initial reference trajectory is completely based on prior knowledge of the environment, the more the adapting reference trajectory deviates from the initial reference trajectory, the less the trajectory will be driven by prior knowledge and the more it is driven by the current movement of the slave. The adaptation rate therefore also represents how reliable the original reference trajectory is. If a model is trained poorly, more emphasis can be put on the slave movement. There are however two possible downsides to adapting reference trajectories based on the movement of the slave.

Firstly it makes the adaptation difficult since the current movement of the slave is a very limited information source for updating the reference trajectory. It is hard to determine what the preferred adaptation is based solely on the state of the slave. In this experiment the preferred adaptation is 'forced' upon the controller by means of tuning the controller for this specific task.

Secondly, a controller with a too high adaptation rate tends to follow the operator more, taking away the benefits of HSC, instead of guiding the operator.

To summarize the discussion on the adaptation rate: a controller with an adaptation rate that is too high may be of less support to the operator and harder to understand by the operator. On the technical side, a high adaptation rate may result in undesired shapes in the reference trajectory due to the limited available information. Contrarily, an adaptation rate that is too low may result in unnecessary high control effort.

5.2 Robustness for varying situations

Another point of discussion is the reaction of the controller to different inputs. The operator can move the slave fast, slow, backward or even stand still during the task execution. These different inputs may require different forms of adaptation and real-world applications should account for this variability in the input. The question that needs

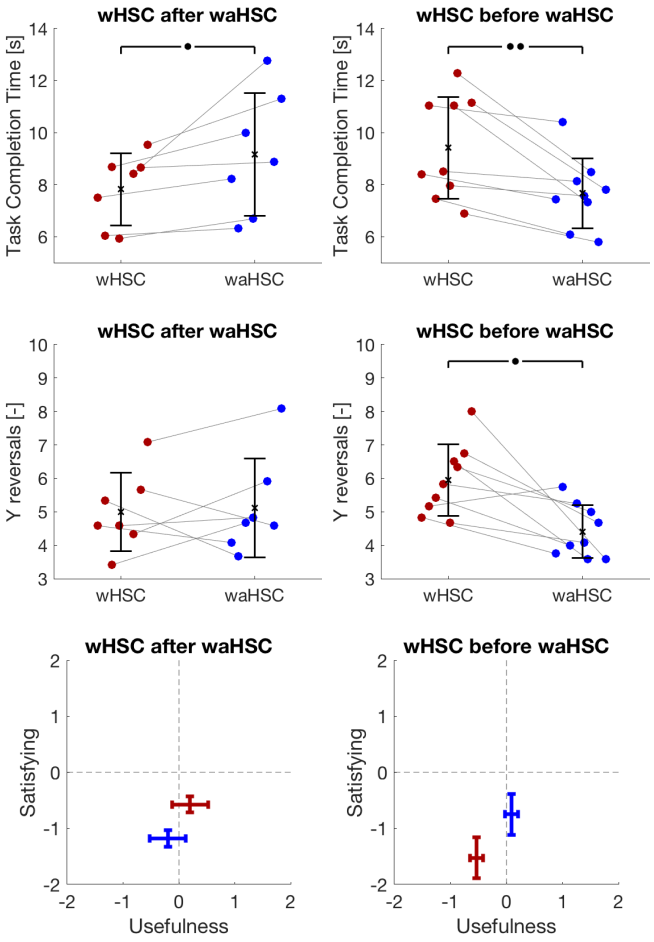


Fig. 10. Results for different order of execution of waHSC and wHSC. The dots (•), (••) respectively denote p-values with $p \leq 0.05, p \leq 0.01$ significance.

addressing is how the controller is expected to adapt in different situations.

As stated in Fitt's law, operators make a trade-off between speed and accuracy during the execution of a task. Operators tend to slow down the movement when they reach a difficult part in the task and increase the speed when less accuracy is required. A constant adaptation therefore might be beneficial considering that the more movement data points are generated for the same movement distance if the movement is slow. The result is that the reference trajectory adapts more during slow movements and less during fast movements. However, in the case where there is no movement at all, the controller is sometimes not expected to adapt. As a result, a different behavior of the controller can be expected for the same operator input. Furthermore, an adaptive controller should also perform properly when the initial reference trajectory is correct. No adaptation is expected from the controller when the reference trajectory is correct.

Summarizing, before an adaptive controller can successfully be implemented, the expected behavior of the controller needs to be clear. Deducting the expected behavior from the slave movement is very hard and will likely only work for precisely defined tasks.

5.3 Limitations and issues

The IGMM based adaptive controller used in this experiment is not without its limits or issues. Most of them are imposed by the way IGMM works. This subsection will cover the limitations first, followed by the issues encountered with the controller.

The first limitation is the very task-specific tuning. The tuning relies strongly on the update frequency and environment. The tuning is required to prevent the adaptation from forming unwanted shapes in the reference trajectory. One of the encountered issues is that the reference trajectory in some circumstances can adapt towards a wrinkle (Fig. 12b). The specific tuning for each task and environment can prevent this to a certain degree.

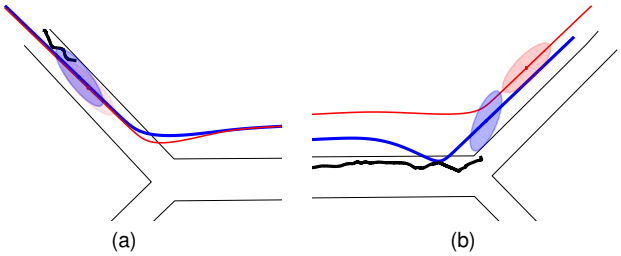


Fig. 12. Plots with initial wrong reference (red) and current adapted reference (blue) showing issues with the IGMM adaptation: (a) adaptation in correct movement, (b) forming of wrinkles.

Secondly, IGMM is not capable of rotating the direction in a reference trajectory. It can only translate parts of the reference trajectory at the locations of the state means. As mentioned in Section 3.2.3, GMR bases the reference trajectory on the most likely (x, y) position for each time-step t, but IGMM does not update the covariances between these variables. In fact, IGMM only adapts the x- and y-related (co)variances for each state with the inability to rotate the reference relative to the state mean as a result.

The last limitation is that the IGMM controller in this research can only handle forward motion. Due to the inclusion of line 4 in the IGMM algorithm, the controller adapts regardless of the input type, which means that when the slave isn't moving the reference trajectory is still adapting towards the slave. This method was sufficient for this experiment since the subjects were instructed to move as fast as possible towards the end point and therefore the input type didn't vary.

In terms of issues, there are some problems that arise which are inherent to the wrong reference trajectory or the core working principles of the used algorithms. The first issue relates to how an IGMM controller works when the slave follows a reference trajectory and there are no conflicts. As discussed earlier, an adaptive controller should also be able to function when the initial reference trajectory is correct. The IGMM based controller in this experiment can't cope with a correct reference trajectory (Fig. 12a). In the event that the slave position is close to the reference trajectory, it's location is by definition in extension of one of

the states in the global GMM. This results in a high value for $p(k|x)$ for state k and therefore state k will move along the reference trajectory towards the slave's position. As a result, the states are adapting even though the slave is on the reference trajectory, while no adaptation is required.

The last issue to discuss is inherent to incorrect trajectories. As became clear when analyzing the raw data in Section 4.1, the location of the wrong reference trajectory relative to the current movement has a significant effect on the movement of the slave. Whenever the movement took the inner turn relative to the reference trajectory, as seen in path 3 in Fig. 9, the direction of the guidance force can change very quickly. This is due to the fact that the direction and magnitude of the guidance force are based on the shortest distance between slave and reference trajectory and that the user is close to the radial center of the corner when taking the inner turn. The implemented look-ahead time, where an estimated future location is used to base the guidance force upon, adds to this effect since the peak in velocity (Fig. 8) results in a wrong estimated future location and thus in incorrect guidance force.

5.4 Implications, recommendations and future work

This research shows that online adaptation of the reference trajectory can be used to reduce conflicts in HSC. However, more research has to be done before an adaptive controller could be generally usable. The use of the designed controller is limited to a very specific task and requires a lot of tuning before it adapts the reference trajectory properly. Based on these limitations and the discussion above, several recommendations can be made for future work.

The controller in this research used a high adaptation rate. As mentioned in the discussion, there are several drawbacks to a high adaptation rate. Teleoperation tasks that are longer—in space and time—allow for a lower adaptation rate. It would therefore be interesting to see what differences an adaptive controller can make in an experiment that involves these longer tasks.

The conducted experiment in this research was not aimed at exploring the effect of the execution order of the different conditions, but these were found to be substantially based on the between-subject analyses. The learning of a movement strategy by the operator could be the cause of the difference in results between different execution orders. Another future research could look into this possible strategy learning and explore the benefits of an adaptive controller for operators with and without a movement strategy.

Finally, a research could be conducted to eliminate some limitations of the designed adaptive controller. Several encountered limitations and issues of the designed controller are related to IGMM. The Direct Update method adapts the reference trajectory in a different way and does not suffer from many of the IGMM related limitations. Some of these limitations, such as the possible wrinkling of the reference trajectory and the intensive tuning to get the desired adaptation behavior, can be circumvented by using the Direct Update method.

6 Conclusion

This paper presented the design and evaluation of a haptic shared controller that is capable of adapting its reference online. The core of the controller consists of a Task Parameterized Gaussian Mixture Model (TP-GMM), which allows for adaptation in different ways and is capable of quickly generating reference trajectories. Literature provided three methods that can adapt an existing Gaussian Mixture Model: Generative Update, Direct Update and an Incremental Gaussian Mixture Model (IGMM). These methods have been compared in a simulation on their computational expense and their adaptation behavior. The Incremental Gaussian Mixture Model showed to be the most promising for implementation in the adapting controller, since it was the fastest method and it only requires the current slave position. The TP-GMM and IGMM have been modified to make them suitable for use in haptic shared control (HSC) for a planar telemanipulated movement task.

The performance of the adapting controller is evaluated with an experiment in which 16 participants performed a trajectory following task on a teleoperation device. Four different conditions were tested: HSC with a correct reference trajectory (cHSC), HSC with a wrong reference trajectory (wHSC), HSC with a wrong reference trajectory that adapts the reference trajectory online (waHSC) and manual operation. The performance of each condition has been experimentally assessed and compared, for which was hypothesized that: waHSC improved performance (H1) and reduced control effort (H2) compared to wHSC, and that waHSC resulted in better performance than manual control (H3). The results led to the following conclusions:

- Within-subject results showed no significant difference in performance between waHSC and wHSC with the used adapting controller. Therefore hypothesis 1 is not accepted.
- waHSC results in less control effort compared to wHSC. This confirms hypothesis 2.
- waHSC results in worse performance relative to manual control. Therefore hypothesis 3 is not accepted.
- When the HSC is based on a perfect reference trajectory, the performance is significantly better and the control effort is significantly lower compared to the other conditions.

Apart from the within-subject results, a between-subject analysis between waHSC and wHSC has been conducted based on the order of execution. This showed decreased task completion time, better self-reported usefulness and more self-reported satisfaction for waHSC relative to wHSC when wHSC was performed before waHSC. When wHSC was performed after waHSC a significant decrease in task completion time was found and wHSC was reported as being more satisfying.

References

[1] T. Imaida, Y. Yokokohji, T. Doi, M. Oda, and T. Yoshikawa, "Ground-space bilateral teleoperation of ETS-VII robot arm by direct bilateral coupling under

7-s time delay condition," *IEEE Trans. Robot. Autom.*, vol. 20, no. 3, pp. 499–511, 2004.

[2] W. Bluethmann, R. Ambrose, M. Diftler, S. Askew, E. Huber, M. Goza, F. Rehnmark, C. Lovchik, and D. Magruder, "Robonaut: A robot designed to work with humans in space," *Auton. Robots*, vol. 14, no. 2-3, pp. 179–197, 2003.

[3] S. Cobos-Guzman, J. Torres, and R. Lozano, "Design of an underwater robot manipulator for a telerobotic system," *Robotica*, vol. 31, no. 6, pp. 945–953, 2013.

[4] M. P. Vecchi and J. A. Salehi, "A Manipulator Work Package for Teleoperation from Unmanned Untethered Vehicles - Current Feasibility and Future Applications," *Int. Adv. Robot. ProgramWorkshop Subsea Robot.*, 1996.

[5] W. Wang and K. Yuan, "Teleoperated manipulator for leak detection of sealed radioactive sources," *Proc. 2004 IEEE, Int. Conf. Robot. Autom.*, vol. 2, pp. 1682—-1687, 2004.

[6] A. M. Okamura, "Methods for haptic feedback in teleoperated robot-assisted surgery," *Ind. Robot An Int. J.*, vol. 31, no. 6, pp. 499–508, 2004.

[7] G. P. Moustris, S. C. Hiridis, K. Deliparaschos, and K. Konstantinidis, "Evolution of autnomous and semi-autonomous robotic surgical systems: a review of the literature," *Int. J. Med. Robot.*, vol. 7, no. April, pp. 375–392, 2011.

[8] A. Kazi, "Operator Performance in Surgical Telemanipulation," *Presence Teleoperators Virtual Environ.*, vol. 10, no. 5, pp. 495–510, 2001.

[9] A. Bolopion and S. Régnier, "A review of haptic feedback teleoperation systems for micromanipulation and microassembly," *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 3, pp. 496–502, 2013.

[10] H. Boessenkool, D. A. Abbink, C. J. M. Heemskerk, F. C. T. Van Der Helm, and J. G. W. Wildenbeest, "A task-specific analysis of the benefit of haptic shared control during telemanipulation," *IEEE Trans. Haptics*, vol. 6, no. 1, pp. 2–12, 2013.

[11] C. Passenberg, A. Peer, and M. Buss, "A survey of environment-, operator-, and task-adapted controllers for teleoperation systems," *Mechatronics*, vol. 20, no. 7, pp. 787–801, 2010.

[12] N. Stefanov, C. Passenberg, A. Peer, and M. Buss, "Design and evaluation of a haptic computer-assistant for telemanipulation tasks," *IEEE Trans. Human-Machine Syst.*, vol. 43, no. 4, pp. 385–397, 2013.

[13] J. Van Oosterhout, J. G. Wildenbeest, H. Boessenkool, C. J. Heemskerk, M. R. De Baar, F. C. Van Der Helm, and D. A. Abbink, "Haptic shared control in telemanipulation: Effects of inaccuracies in guidance on task execution," *IEEE Trans. Haptics*, vol. 8, no. 2, pp. 164–175, 2015.

[14] C. Passenberg, R. Groten, A. Peer, and M. Buss, "Towards Real-Time Haptic Assistance Adaptation Optimizing Task Performance and Human Effort," *World Haptics Conf. (WHC), 2011 IEEE*, pp. 155–160, 2011.

[15] Y. Li, K. P. Tee, W. L. Chan, R. Yan, Y. Chua, and D. K. Limbu, "Continuous Role Adaptation for Human Robot Shared Control," *IEEE Trans. Robot.*, vol. 31, no. 3, pp. 672–681, 2015.

[16] C. Passenberg, A. Glaser, and A. Peer, "Exploring the design space of haptic assistants: The assistance policy module," *IEEE Trans. Haptics*, vol. 6, no. 4, pp. 440–452, 2013.

[17] D. A. Abbink, M. Mulder, and E. R. Boer, "Haptic shared control: Smoothly shifting control authority?" *Cogn. Technol. Work*, vol. 14, no. 1, pp. 19–28, 2012.

[18] A. W. De Jonge, J. G. W. Wildenbeest, H. Boessenkool, and D. A. Abbink, "The Effect of Trial-by-Trial Adaptation on Conflicts in Haptic Shared Control for Free-Air Teleoperation Tasks," *IEEE Trans. Haptics*, vol. 9, no. 1, pp. 111–120, 2016.

[19] S. J. Lederman and R. L. Klatzky, "Haptic perception: A tutorial," *Attention, Perception, Psychophys.*, vol. 71, no. 7, pp. 1439–1459, 2009.

[20] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intell. Serv. Robot.*, vol. 9, no. 1, pp. 1–29, 2016.

[21] I. Havoutis and S. Calinon, "Learning assistive teleoperation behaviors from demonstration," in *Int. Symp. Safety, Secur. Rescue Robot.*, 2016.

[22] F. Hoeckx, "The Effect of Individualised Haptic Guidance Paths for Free-Air Teleoperation Tasks," Delft University of Technology, Tech. Rep., 2016.

[23] M. Najafi, K. Adams, and M. Tavakoli, "Robotic Learning From Demonstration of Therapist's Time-Varying Assistance to a Patient in Trajectory-Following Tasks," *IEEE Int. Conf. Rehabil. Robot.*, 2017.

[24] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *J. Artif. Intell. Res.*, vol. 4, pp. 129–145, 1996.

[25] S. Calinon and A. Billard, "Learning of Gestures by Imitation in a Humanoid Robot," *Imitation Soc. Learn. Robot. Humans Anim. Behav. Soc. Commun. Dimens.*, pp. 153–177, 2007.

[26] P. Engel and M. Heinen, "Incremental learning of multivariate Gaussian mixture models," *Adv. Artif. Intell. 2010*, pp. 82–91, 2011. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-16138-4_9

[27] G. A. V. Christiansson, "Hard Master, Soft Slave Haptic Teleoperation," Ph.D. dissertation, Delft University of Technology, 2007.

[28] J. G. W. Wildenbeest, D. A. Abbink, C. J. M. Heemskerk, F. C. T. Van Der Helm, and H. Boessenkool, "The impact of haptic feedback quality on the performance of teleoperated assembly tasks," *IEEE Trans. Haptics*, vol. 6, no. 2, pp. 242–252, 2013.

[29] B. A. Forsyth and K. E. MacLean, "Predictive Haptic Guidance: Intelligent User Assistance for the Control of Dynamic Tasks," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 1, pp. 103–113, 2006.

[30] J. D. Van Der Laan, A. Heino, and D. De Waard, "A simple procedure for the assessment of acceptance of advanced transport telematics," *Transp. Res. Part C Emerg. Technol.*, vol. 5, no. 1, pp. 1–10, 1997.

A

TP-GMM overview and implementation

This appendix will describe how TP-GMM works and how it is implemented in MATLAB.

A.1. Overview of TP-GMM

Task Parameterized Gaussian Mixture Models (TP-GMM) allow for the creation of a Gaussian Mixture Model (GMM) that is specific to a certain task. Gaussian Mixture Regression (GMR) can be used on the created GMM to generate a regression line, which in turn can be used as a reference trajectory in Haptic Shared Control. Figure A.1 gives an overview of the steps that will be discussed in section A.2. Finally, section A.3 will provide a visualization of each step.

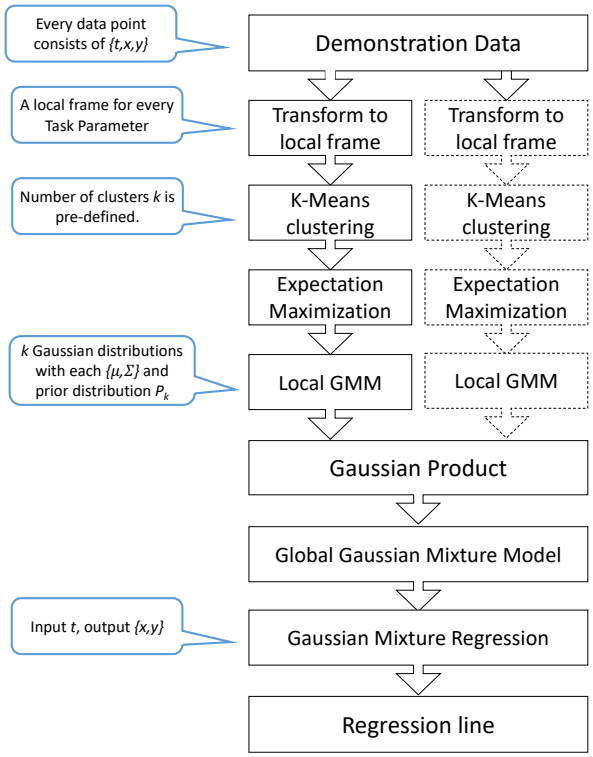


Figure A.1: Overview of TP-GMM.

A.2. TP-GMM steps

A.2.1. Parameter notation

Before the implementations and equations will be discussed, several parameters have to be assigned and explained. Table A.1 shows the parameters and values used for the model, along with their counters used in the equations and MATLAB code.

Table A.1: Annotation of parameters and values of the model

Variable	Annotation	Counter	Value	Unit
Dimension	N	-	2	[-]
States	K	k	4	[-]
Reference frames	P	p	2	[-]
Demonstrations	M	m	4	[-]
Data points per demo	T	t	200	[-]
Data points total	D	d	$D = MT$	[-]
K-means iterations	-	-	20	[-]
EM threshold	C_1	-	10^{-5}	[-]

Table A.2 shows the parameters used in the equations and MATLAB code, along with their dimension and explanation.

Table A.2: Explanation and description of equation parameters

Symbol	MATLAB variable	Dimension	Description
$\boldsymbol{A}_{p,m}$	`s(m).p(p).A`	$N \times N$	Rotation matrix
$\boldsymbol{b}_{p,m}$	`s(m).p(p).b`	$N \times 1$	Translation matrix
$\boldsymbol{\xi}_{t,m}$	`s(m).Data0(:,t)`	$N \times T$	Data point in Global matrix
$\boldsymbol{X}$	`Data`	$N \times P \times D$	Local data tensor
$\boldsymbol{\mu}_{k,p}$	`model.Mu(:,p,k)`	$N \times P \times K$	Means of model matrix
$\boldsymbol{\Sigma}_{k,p}$	`model.Sigma(:,:,p,k)`	$N \times N \times P \times K$	Covariance of model matrix
π_k	`model.Priors(k)`	$K \times 1$	Model priors
$h_k(\boldsymbol{\xi}_t)$	`H(i,t)`	$K \times T$	Activation weight for each data point

A.2.2. Demonstrations

Demonstrations are stored in the ***s*** (*s*ample) struct. This struct consists of the movement data, a variable depicting the number of data points (`nbData`) and the demonstration specific task parameters. Task parameters consist of a rotation matrix A and translation matrix b that determine the location and orientation of the local reference frame.

$$\boldsymbol{\xi}_m = \begin{bmatrix} t_1 & \cdots & t_t \\ x_1 & \cdots & x_t \\ y_1 & \cdots & y_t \end{bmatrix} \tag{A.1}$$

The $\boldsymbol{s}_m$ matrices contain coordinates of the examples in the global reference frame (see A.1), which has to be converted to data relative to the reference frames for future calculations. This is done using the following formula, considering time elements $t_1 \dots t_t$ have been removed:

$$\boldsymbol{X}_{p,t} = \boldsymbol{A}_p^{-1}(\boldsymbol{\xi}_{t,m} - \boldsymbol{b}_p) \tag{A.2}$$

The resulting data matrices for all p are then combined into a data tensor $\boldsymbol{X}$ for easy computing with MATLAB.

A.2.3. K-means

The K-means algorithm divides the data into a set amount (K) clusters based on the spacing between the data points. Before the clustering is done, the data tensor is flattened ($N \times P \times D$ to $N \times PD$) so it can be handled at once by the K-means algorithm. For example a data tensor with $N = 2$, $M = 4$:

$$\boldsymbol{X} = \begin{bmatrix} x_{1,1} & \cdots & x_{p,1} \\ y_{1,1} & \cdots & y_{p,1} \end{bmatrix}\Bigg]\Bigg]\Bigg]_m = \begin{bmatrix} x_{1,1} & \cdots & x_{1,m} \\ y_{1,1} & \cdots & y_{1,m} \\ \vdots & \ddots & \vdots \\ x_{p,1} & \cdots & x_{p,m} \\ y_{p,1} & \cdots & y_{p,m} \end{bmatrix} \tag{A.3}$$

The K-means algorithm is based on the following formula:

$$\boldsymbol{\mu}_k = \arg\min_{\mu} \sum_{k=1}^{K} \sum_{d=1}^{D} ||\boldsymbol{x}_{k,d} - \boldsymbol{\mu}_k||^2 \tag{A.4}$$

When the K-means algorithm (`kmeansClustering.m` [2]) is finished, it returns a list that assigns each data point to one of the states. The calculated distribution can be a local optimum, since the starting means are generated randomly. Therefore the code is adjusted to chose the distribution over the states with the lowest cumulative distance over 20 iterations, increasing the likelihood of finding the global optimum. This is done for consistency in the results when running the code.

Additionally, the list is used to calculate the means $\boldsymbol{\mu}_{k,p}$ and covariances $\boldsymbol{\Sigma}_{k,p}$. This is however optional, since in this application K-means is used as an initial distribution for Expectation Maximization (EM). Mean matrix $\boldsymbol{\mu}_{k,p}$ and covariance matrix $\boldsymbol{\Sigma}_{k,p}$ will therefore be overwritten by the EM algorithm.

A.2.4. Expectation Maximization

Expectation Maximization consists of two steps, namely the expectation and maximization step. In the so-called E-step the probability is calculated that a data point belongs to a certain distribution. This is done for all the data points. Based on this probability, new means and covariances are calculated in the M-step. Each point has a probability that it belongs to a certain distribution, so based upon this each data point might contribute more in the mean calculation of one distribution than the other.

The steps in the expectation maximization algorithm are as follows:

E-step:

$$h_k^u(\boldsymbol{\xi}_d) = \frac{\pi_k^u \mathcal{N}(\boldsymbol{\xi}_d \mid \boldsymbol{\mu}_k^u, \boldsymbol{\Sigma}_k^u)}{\sum_{k=1}^{K} \pi_k^u \mathcal{N}(\boldsymbol{\xi}_d \mid \boldsymbol{\mu}_k^u, \boldsymbol{\Sigma}_k^u)} \tag{A.5}$$

$$E_k^u = \sum_{d=1}^{D} h_k^u(\boldsymbol{\xi}_d) \tag{A.6}$$

M-step:

$$\pi_k^{u+1} = \frac{E_k^u}{D} \tag{A.7}$$

$$\boldsymbol{\mu}_k^{u+1} = \frac{\sum_{d=1}^{D} h_k^u(\boldsymbol{\xi}_d)\boldsymbol{\xi}_d}{E_k^u} \tag{A.8}$$

$$\boldsymbol{\Sigma}_k^{u+1} = \frac{\sum_{d=1}^{D} h_k^u(\boldsymbol{\xi}_d)(\boldsymbol{\xi}_d - \boldsymbol{\mu}_k^{u+1})(\boldsymbol{\xi}_d - \boldsymbol{\mu}_k^{u+1})^\top}{E_k^u} \tag{A.9}$$

The EM algorithm is stopped whenever the increase of the total log-likelihood drops below a certain percentile threshold C_1, which is calculated as follows:

$$\mathcal{L}^{u+1}(\xi) = \sum_{d=1}^{D} \log\left(\sum_{k=1}^{K} \pi_k^{u+1} \mathcal{N}(\boldsymbol{\xi}_d \mid \boldsymbol{\mu}_k^{u+1}, \boldsymbol{\Sigma}_k^{u+1}) \right) \tag{A.10}$$

$$\frac{\mathcal{L}^{u+1}(\xi)}{\mathcal{L}^{u}(\xi)} < C_1 \tag{A.11}$$

When likelihood doesn't increase any further, and thus a local optimum has been reached, the EM algorithms returns the $\boldsymbol{\mu}_{k,p}$ and $\boldsymbol{\Sigma}_{k,p}$ of the local optimum. Furthermore, the model priors π_k are now known, which represent the fraction of the data points assigned to a specific state and show the probability that a random data point belongs to a specific state.

The code written by Calinon [2], `EM_tensorGMM.m`, is used for this application. The value for C_1 in the original `EM_tensorGMM.m` has been adopted for this research.

A.2.5. Gaussian product

The model describes the location and variation of the states relative to the local reference (or 'frame'). To create a global GMM they need to be placed in a global reference frame based on their task parameters. Transforming the location and orientation of the mean and covariances from the local frame to the global frame is done by using the task parameters. This transformation is done using the following equations:

$$\hat{\boldsymbol{\mu}}_{k,p} = \boldsymbol{A}_p \boldsymbol{\mu}_{k,p} + \boldsymbol{b}_p \tag{A.12}$$

$$\hat{\boldsymbol{\Sigma}}_{k,p} = \boldsymbol{A}_p \boldsymbol{\Sigma}_{k,p} \boldsymbol{A}_p^\top \tag{A.13}$$

When all the local GMMs are transformed to the global frame, they can be multiplied with each other by using the Gaussian Product. The Gaussian Product is defined as:

$$\hat{\boldsymbol{\Sigma}}_k = \Big(\sum_{p=1}^{P} \hat{\boldsymbol{\Sigma}}_{k,p}^{-1} \Big)^{-1} \tag{A.14}$$

$$\hat{\boldsymbol{\mu}}_k = \hat{\boldsymbol{\Sigma}}_k \sum_{p=1}^{P} \hat{\boldsymbol{\Sigma}}_{k,p}^{-1} \hat{\boldsymbol{\mu}}_{k,p} \tag{A.15}$$

The combined situation specific states are known as a Gaussian Mixture Model (GMM).

A.2.6. Gaussian Mixture Regression

Now the situation specific GMM is known. With this information the regression data, with the respective regression line, can be generated using Gaussian Mixture Regression (GMR). This is done in the `GMR.m` file. GMR estimates $\widetilde{\mu}$ and $\widetilde{\Sigma}$ for each time step. This is retrieved by calculating the following probability distribution:

$$\mathcal{P}(\boldsymbol{\xi}_t^O \mid \boldsymbol{\xi}_t^I) \sim \sum_{k=1}^{K} h_k(\boldsymbol{\xi}_t^I) \mathcal{N}\big(\hat{\boldsymbol{\mu}}_k^O(\boldsymbol{\xi}_t^I), \hat{\boldsymbol{\Sigma}}_{\boldsymbol{k}}^O\big) \tag{A.16}$$

$$\text{Where:} \quad \hat{\boldsymbol{\mu}}_k^O(\boldsymbol{\xi}_t^I) = \boldsymbol{\mu}_k^O + \boldsymbol{\Sigma}_k^{OI} (\boldsymbol{\Sigma}_k^I)^{-1} (\boldsymbol{\xi}_t^I - \boldsymbol{\mu}_k^I) \tag{A.17}$$

$$\hat{\boldsymbol{\Sigma}}_k^O = \boldsymbol{\Sigma}_k^O - \boldsymbol{\Sigma}_k^{OI} (\boldsymbol{\Sigma}_k^I)^{-1} \boldsymbol{\Sigma}_k^{IO} \tag{A.18}$$

However, the resulting probability distribution is multi-modal. The different states are summed rather than 'merged', which results in the possibility for local optima and is therefore not suitable for GMR. Instead, an approximation to this distribution is used:

$$\mathcal{P}(\boldsymbol{\xi}_t^O \mid \boldsymbol{\xi}_t^I) = \mathcal{N}\big(\hat{\boldsymbol{\xi}}_t^O \mid \hat{\boldsymbol{\mu}}_t^O, \hat{\boldsymbol{\Sigma}}_t^O\big) \tag{A.19}$$

$$\text{Where:} \quad \hat{\boldsymbol{\mu}}_t^O = \sum_{k=1}^{K} h_k(\boldsymbol{\xi}_t^I) \hat{\boldsymbol{\mu}}_k^O(\boldsymbol{\xi}_t^I) \tag{A.20}$$

$$\hat{\boldsymbol{\Sigma}}_t^O = \sum_{k=1}^{K} h_k(\boldsymbol{\xi}_t^I) \big(\hat{\boldsymbol{\Sigma}}_k^O + \hat{\boldsymbol{\mu}}_k^O(\boldsymbol{\xi}_t^I) \hat{\boldsymbol{\mu}}_k^O(\boldsymbol{\xi}_t^I)^\top\big) - \hat{\boldsymbol{\mu}}_t^O (\hat{\boldsymbol{\mu}}_t^O)^\top \tag{A.21}$$

In equation A.21 an optional regularization term can be added to the diagonal of the matrix to ensure the stability of the algorithm.

The h_k term in the equations is the activation weight, depicting the weight of each state at each data point and ensuring a smooth transition between the states by doing so.

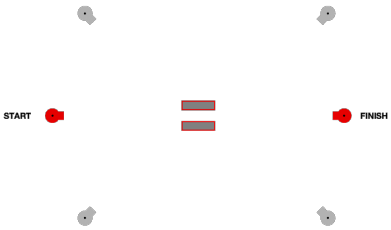
$$h_k(\boldsymbol{\xi}_t^I) = \frac{\pi_k \mathcal{N}\big(\boldsymbol{\xi}_t^I \mid \boldsymbol{\mu}_k^I, \boldsymbol{\Sigma}_k^I\big)}{\sum_{k=1}^{K} \pi_k \mathcal{N}\big(\boldsymbol{\xi}_t^I \mid \boldsymbol{\mu}_k^I, \boldsymbol{\Sigma}_k^I\big)} \tag{A.22}$$

Where the conditional probability is calculated using the classical formula:

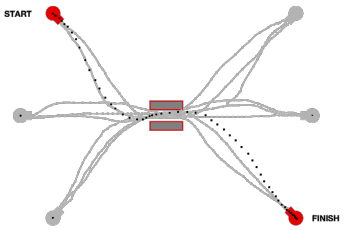
$$\mathcal{N}(\boldsymbol{\xi}_t^I \mid \boldsymbol{\mu}_k^I, \boldsymbol{\Sigma}_k^I) = \frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}_k^I|}} e^{-\frac{1}{2}(\boldsymbol{\xi}_t^I - \boldsymbol{\mu}_k^I)^\top \boldsymbol{\Sigma}_k^I (\boldsymbol{\xi}_t^I - \boldsymbol{\mu}_k^I)} \tag{A.23}$$

A.3. Visualization of TP-GMM steps

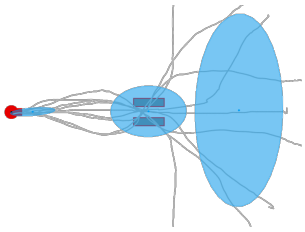
All the TP-GMM steps are visualized below. Steps A.2c to A.2e show the clustering for each of the local frames. Steps A.2g and A.2h show the Gaussian Product with the resulting global GMM in step A.2i. GMR is applied in steps A.2j and A.2k, with the resulting reference trajectory in step A.2l



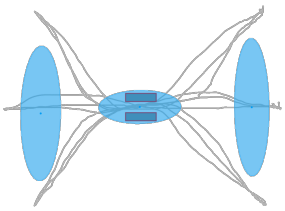
(a) Training environment



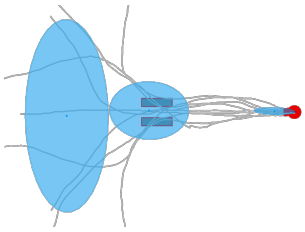
(b) Demonstrations complete



(c) Local GMM for start TP

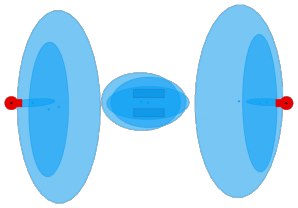


(d) Local GMM for middle TP

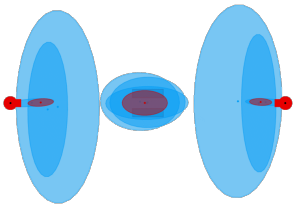


(e) Local GMM for end TP

(f) Target Environment



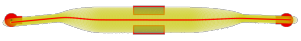
(g) Local GMMs overlay for target environment



(h) Gaussian Product

(i) Global GMM

(j) Gaussian Mixture Regression



(k) Regression line

(l) Resulting reference trajectory

Figure A.2: Visualization of all the TP-GMM steps

A.4. Nesting of MATLAB functions

The steps described in the appendix is all implemented in MATLAB. Table A.2 shows the parameters with their corresponding equivalents in MATLAB. Table 1 shows the nesting of the mentioned MATLAB functions together with the location of the discussed equations.

Algorithm 1 Nesting of MATLAB functions with corresponding described equations

Require: M Demonstrations with each a $K \times T$ data matrix

function MAIN.M
  Set parameters: K, P, A, b
  Retrieve parameters from demonstrations: N, M, T ▷ A.1
  for every M **do**
    for every P **do**
      $\boldsymbol{X}_{p,t} = \boldsymbol{A}_p^{-1}(\boldsymbol{\xi}_{t,m} - \boldsymbol{b}_p)$ ▷ A.2
    end for
  end for
  function INIT_TENSORGMM_KMEANS.M
    Flatten data tensor $\boldsymbol{X}$ for K-means ▷ A.3
    function KMEANS_CLUSTERING.M($\boldsymbol{X}, K$)
      for every P **do**
        $\boldsymbol{\mu}_{k,p} = \arg\min_{\mu} \sum_{k=1}^{K} \sum_{d=1}^{D} ||\boldsymbol{x}_{k,d} - \boldsymbol{\mu}_{k,p}||^2$ ▷ A.4
      end for
      return $\boldsymbol{\mu}_{k,p}$
    end function
    Calculate priors π_k and covariance $\boldsymbol{\Sigma}_{k,p}$ based on $\boldsymbol{X}, \boldsymbol{\mu}_{k,p}$
    return $\boldsymbol{\mu}_{k,p}, \boldsymbol{\Sigma}_{k,p}, \pi_k$
  end function
  function EM_TENSORGMM.M($\boldsymbol{X}, \boldsymbol{\mu}_{k,p}, \boldsymbol{\Sigma}_{k,p}, \pi_k$)
    for every P **do**
      while $\frac{\mathcal{L}^{u+1}(\xi)}{\mathcal{L}^{u}(\xi)} < C_1$ **do** ▷ A.11
        $h_k^u(\boldsymbol{\xi}_d) = \frac{\pi_k^u \mathcal{N}\left(\boldsymbol{\xi}_d | \boldsymbol{\mu}_k^u, \boldsymbol{\Sigma}_k^u\right)}{\sum_{k=1}^{K} \pi_k^u \mathcal{N}\left(\boldsymbol{\xi}_d | \boldsymbol{\mu}_k^u, \boldsymbol{\Sigma}_k^u\right)}$ ▷ A.5
        $E_k^u = \sum_{d=1}^{D} h_k^u(\boldsymbol{\xi}_d)$ ▷ A.6
        $\pi_k^{u+1} = \frac{E_k^u}{D}$ ▷ A.7
        $\boldsymbol{\mu}_k^{u+1} = \frac{\sum_{d=1}^{D} h_k^u(\boldsymbol{\xi}_d)\boldsymbol{\xi}_d}{E_k^u}$ ▷ A.8
        $\boldsymbol{\Sigma}_k^{u+1} = \frac{\sum_{d=1}^{D} h_k^u(\boldsymbol{\xi}_d)(\boldsymbol{\xi}_d - \boldsymbol{\mu}_k^{u+1})(\boldsymbol{\xi}_d - \boldsymbol{\mu}_k^{u+1})^\top}{E_k^u}$ ▷ A.9
        $\mathcal{L}^{u+1}(\xi) = \sum_{d=1}^{D} \log\left(\sum_{k=1}^{K} \pi_k^{u+1} \mathcal{N}\left(\boldsymbol{\xi}_d \mid \boldsymbol{\mu}_k^{u+1}, \boldsymbol{\Sigma}_k^{u+1}\right)\right)$ ▷ A.10
      end while
    end for
    return $\boldsymbol{\mu}_{k,p}, \boldsymbol{\Sigma}_{k,p}, \pi_k$
  end function
  function PRODUCTTPGMM0.M($\boldsymbol{\mu}_{k,p}, \boldsymbol{\Sigma}_{k,p}, \boldsymbol{A}_p, \boldsymbol{b}_p$)
    $\hat{\boldsymbol{\mu}}_{k,p} = \boldsymbol{A}_p \boldsymbol{\mu}_{k,p} + \boldsymbol{b}_p$ ▷ A.12
    $\hat{\boldsymbol{\Sigma}}_{k,p} = \boldsymbol{A}_p \boldsymbol{\Sigma}_{k,p} \boldsymbol{A}_p^\top$ ▷ A.13
    $\hat{\boldsymbol{\Sigma}}_k = \left(\sum_{p=1}^{P} \hat{\boldsymbol{\Sigma}}_{k,p}^{-1}\right)^{-1}$ ▷ A.14
    $\hat{\boldsymbol{\mu}}_k = \hat{\boldsymbol{\Sigma}}_k \sum_{p=1}^{P} \hat{\boldsymbol{\Sigma}}_{k,p}^{-1} \hat{\boldsymbol{\mu}}_{k,p}$ ▷ A.15
    return $\hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\Sigma}}_k$
  end function
  function GMR.M
    $h_k(\boldsymbol{\xi}_t^I) = \frac{\pi_k \mathcal{N}\left(\boldsymbol{\xi}_t^I | \boldsymbol{\mu}_k^I, \boldsymbol{\Sigma}_k^I\right)}{\sum_{k=1}^{K} \pi_k \mathcal{N}\left(\boldsymbol{\xi}_t^I | \boldsymbol{\mu}_k^I, \boldsymbol{\Sigma}_k^I\right)}$ ▷ A.22
    function GAUSSPDF.M
      $\mathcal{N}(\boldsymbol{\xi}_t^I \mid \boldsymbol{\mu}_k^I, \boldsymbol{\Sigma}_k^I) = \frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}_k^I|}} e^{-\frac{1}{2}(\boldsymbol{\xi}_t^I - \boldsymbol{\mu}_k^I)^\top \boldsymbol{\Sigma}_k^I (\boldsymbol{\xi}_t^I - \boldsymbol{\mu}_k^I)}$ ▷ A.23
    end function
    $\hat{\boldsymbol{\mu}}_t^O = \sum_{k=1}^{K} h_k(\boldsymbol{\xi}_t^I) \hat{\boldsymbol{\mu}}_k^O(\boldsymbol{\xi}_t^I)$ ▷ A.20

    $\hat{\boldsymbol{\Sigma}}_t^O = \sum_{k=1}^{K} h_k(\boldsymbol{\xi}_t^I) \left(\hat{\boldsymbol{\Sigma}}_k^O + \hat{\boldsymbol{\mu}}_k^O(\boldsymbol{\xi}_t^I) \hat{\boldsymbol{\mu}}_k^O(\boldsymbol{\xi}_t^I)^\top\right) - \hat{\boldsymbol{\mu}}_t^O (\hat{\boldsymbol{\mu}}_t^O)^\top$ ▷ A.21
    $\mathcal{P}(\boldsymbol{\xi}_t^O \mid \boldsymbol{\xi}_t^I) = \mathcal{N}\left(\hat{\boldsymbol{\xi}}_t^O \mid \hat{\boldsymbol{\mu}}_t^O, \hat{\boldsymbol{\Sigma}}_t^O\right)$ ▷ A.19
  end function
end function

B

Reference adaptation methods

As discussed in the previous chapter, the global GMM will be generated using TP-GMM. The adaptation methods described in this chapter all adapt the resulting reference trajectory by making changes in one of the TP-GMM steps. This chapter explains the working principle of each method, how they are compared and which one is most suitable for adapting the reference online.

B.1. Three adaptation methods

A total of 3 different adaptation methods have been investigated: Generative Update, Direct Update and IGMM. Each of the methods adapts the reference trajectory on a different 'level' in the process of generating a reference trajectory. Generative Update works on the deepest level by adapting the demonstration data, Direct Update uses an alternate form of Expectation Maximization to result in an adapted reference trajectory and IGMM adjusts the global GMM to change the reference trajectory. An overview of can be found in Figure B.1.

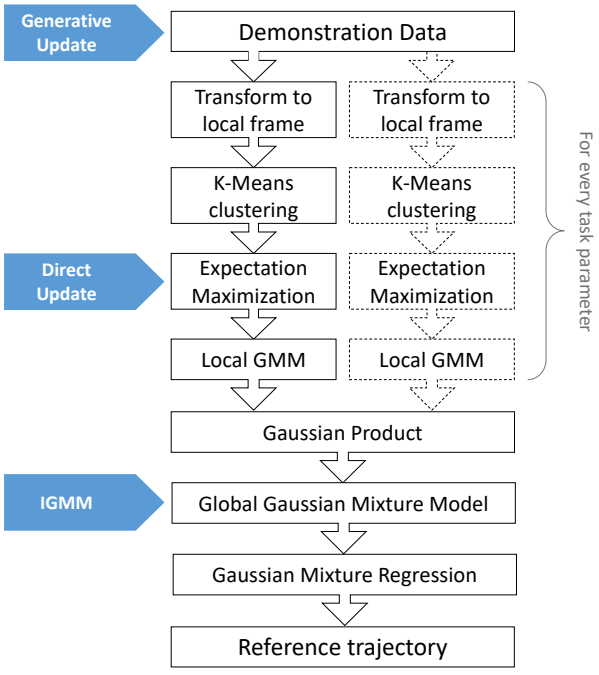


Figure B.1: Overview of TP-GMM with the locations where the different methods adapt.

B.1.1. Generative update

Generative Update (GU) is a GMM update method that has been introduced by Calinon and Billard [3]. This method works on the deepest level in TP-GMM and therefore requires the most time for each update. The update steps in GU work as follows:

1. The current movement of the operator is logged n_1.

2. An existing reference trajectory is used to sample a set of new demonstration data points n_2 that looks similar to the existing reference trajectory.

3. The new combined data set $n = n_1 + n_2$ is used to create a new reference trajectory.

The speed at which this method adapts to the new movements can be determined by changing the ratio of new/sampled data defined by $\alpha \in [0;1]$ in $n\alpha = n_1$. A more conservative adaptation setting requires more sampled data, with the added computational expenses, while quicker adaptation requires less sampled data. Less sampled data also results in a worse representation of the original model.

Another downside of this method is the randomness in sampling the data. Since the combined data set will be slightly different for each update, the resulting reference trajectory will be different as well. This can lead to shock-wise adaptation.

B.1.2. Direct Update

The Direct Update method has also been introduced by Calinon and Billard [3]. This method works by using a modified Expectation Maximization procedure:

E-step:

$$h_k^u(\boldsymbol{\xi}_d) = \frac{\pi_k^u \mathcal{N}\left(\boldsymbol{\xi}_d \mid \boldsymbol{\mu}_k^u, \boldsymbol{\Sigma}_k^u\right)}{\sum_{k=1}^{K} \pi_k^u \mathcal{N}\left(\boldsymbol{\xi}_d \mid \boldsymbol{\mu}_k^u, \boldsymbol{\Sigma}_k^u\right)} \tag{B.1}$$

$$E_k^u = \sum_{d=1}^{D} h_k^u(\boldsymbol{\xi}_d) \tag{B.2}$$

M-step:

$$\pi_k^{u+1} = \frac{E_k^u + E_k^0}{D + D^0} \tag{B.3}$$

$$\boldsymbol{\mu}_k^{u+1} = \frac{\sum_{d=1}^{D} h_k^u(\boldsymbol{\xi}_d)\boldsymbol{\xi}_d + E_k^0 \boldsymbol{\mu}_k^0}{E_k^u + E_k^0} \tag{B.4}$$

$$\boldsymbol{\Sigma}_k^{u+1} = \frac{\sum_{d=1}^{D} h_k^u(\boldsymbol{\xi}_d)(\boldsymbol{\xi}_d - \boldsymbol{\mu}_k^{u+1})(\boldsymbol{\xi}_d - \boldsymbol{\mu}_k^{u+1})^\top}{E_k^u + E_k^0} \tag{B.5}$$

$$+ \frac{E_k^0(\boldsymbol{\Sigma}_k^0 + (\boldsymbol{\mu}_k^0 - \boldsymbol{\mu}_k^{u+1})(\boldsymbol{\mu}_k^0 - \boldsymbol{\mu}_k^{u+1})^\top)}{E_k^u + E_k^0} \tag{B.6}$$

The expectation step is the the same as in the regular EM algorithm (A.2.4). The maximization step is different; the amount of points and the summed expectancy of the current movement and demonstration movements are used as a weight ratio in the M-step. The adaptation rate can therefore be tuned by changing the weight ratio. This can be achieved by replacing the existing values for E_k^0 and D^0 with E_k^{0*} and D^{0*} as follows:

$$E_k^{0*} = \alpha E_k^0 \tag{B.7}$$

$$D^{0*} = \alpha D^0 \tag{B.8}$$

Where α is the tunable adaptation rate.

B.1.3. Incremental Gaussian Mixture Model

Introduced by Engel and Heinen [4], the Incremental Gaussian Mixture Model (IGMM) adapts the global GMM. The algorithm consists of the following equations:

$$sp_k(t) = sp_k(t-1) + p(k|\boldsymbol{x}) \tag{B.9}$$

$$\omega_k = \frac{p(k|\boldsymbol{x})}{sp_k} \tag{B.10}$$

$$\boldsymbol{e}_k = \boldsymbol{x} - \boldsymbol{\mu}_k(t-1) \tag{B.11}$$

$$\Delta\boldsymbol{\mu}_k = \omega_k \boldsymbol{e}_k \tag{B.12}$$

$$\boldsymbol{\mu}_k(t) = \boldsymbol{\mu}_k(t-1) + \Delta\boldsymbol{\mu}_k \tag{B.13}$$

$$\boldsymbol{e}_k = \boldsymbol{x} - \boldsymbol{\mu}_k(t) \tag{B.14}$$

$$\boldsymbol{\Sigma}_k(t) = (1-\omega_k)\boldsymbol{\Sigma}_k(t-1) + \omega_k \boldsymbol{e}_j^* \boldsymbol{e}_j^{*T} - \Delta\boldsymbol{\mu}_k \Delta\boldsymbol{\mu}_k^T \tag{B.15}$$

This adaptation method adapts every state in the global GMM separately. The first step in the process is determining the likelihood $p(k|\boldsymbol{x})$ that the current slave position belongs to the state that is being adapted. This value is added to the saturation parameter $sp_k(t)$. The likelihood is divided by the saturation parameter to retrieve ω_k, which represents the confidence in the current slave position. A higher value for ω_k will result in a bigger adaptation of the global GMM. The saturation parameter $sp_k(t)$ prevents the state from trailing the slave, since the state is less likely to adapt when the saturation parameter fills up. The actual displacement of the state $\Delta\boldsymbol{\mu}_k$ is calculated by multiplying ω_k with the error between the state and the current position of the slave. Finally, the covariance of the state is adapted using equation B.15, which is derived from the Gauss-Wishart distribution.

The adaptation rate of the IGMM can be changed by applying a gain to ω_k. This can be implemented by changing the value for ω_k with ω_k^*, which is defined as follows:

$$\omega_k^* = \alpha\omega_k \tag{B.16}$$

The other modifications mentioned in the paper haven't been implemented during the simulation tests.

B.2. Simulation

The behavior of the methods is tested in a simulation. User movements have been recorded in a training environment by logging the location of the computer cursor. The training environment (Figure B.2) consisted of a starting point, an obstacle and an end point. The starting point and end point had 3 variations each, so a total of 9 different paths from start to end were possible.

The logged data is re-sampled so all the demonstrations had the same length of 100 data points. The logged data is then fed to the algorithms point by point in a for-loop to simulate a real-time movement.

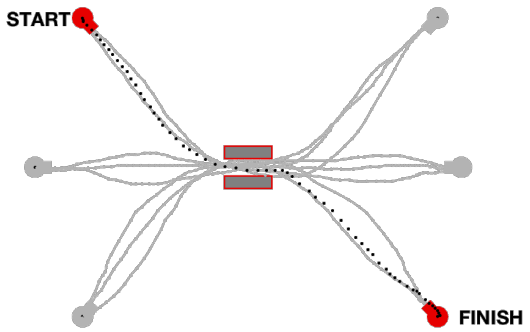


Figure B.2: Training environment. A total of 9 paths were demonstrated.

Two simulation experiments have been conducted. One experiment was aimed at exploring the computational expense of each method. The other investigated the behavior of the different adaptation methods.

B.2.1. Adaptation behavior

The adaption behavior of each method is tested on two different aspects. One aspect is the behavior for different changes in the environment. The methods are tested in an environment where the obstacle is rotated (Figure B.3) and an environment where the obstacle and end point are translated (Figure B.4).

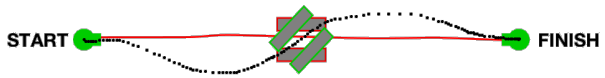


Figure B.3: Simulation environment with a rotated obstacle.

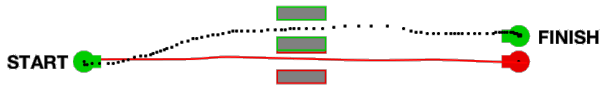


Figure B.4: Simulation environment with a translated obstacle and end point.

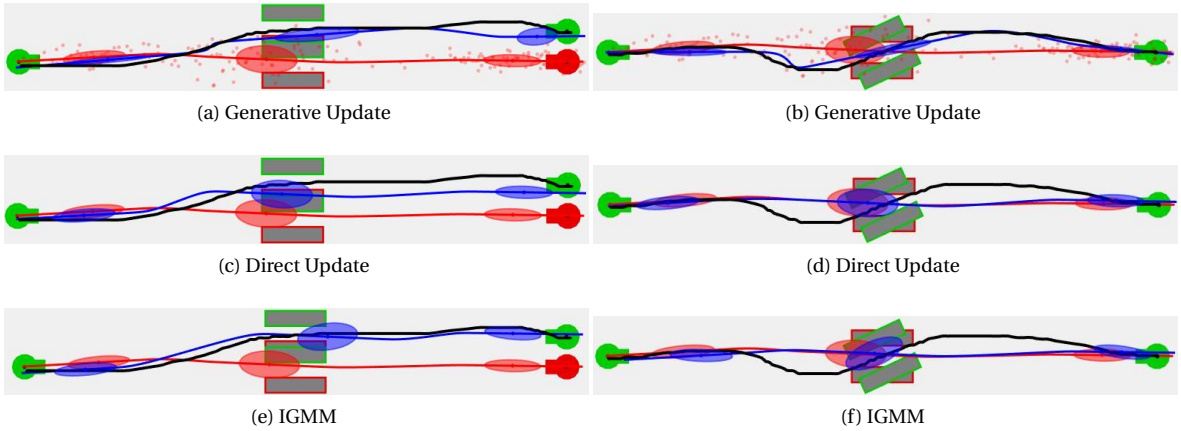
The second aspect is the behavior related to the movement speed of the operator. This behavior is tested in for different constant speeds and for a varying speed. The different constant speeds are simulated by re-sampling an existing movement trajectory to 200 and 50 data points, respectively the double and half of the original number of data points. If a constant logging frequency of the operator's movement is assumed, the movement trajectory with 50 data points represents a faster movement than the movement trajectory with 100 data points. The operator movement with varying speed is recorded separately and re-sampled to 100 data points. The speed simulations are conducted in the simulation environment of Figure B.4. The conclusions on the behavior of the different adaptation methods can be found in Table B.1.

	Generative Update	**Direct Update**	**IGMM**
Type of adaptation	Complete new model	Relative to original model	Relative to last update
Handles varying movement speeds	No	Yes	Yes
Handles different movement speeds	No	Frequency should be the same as in training	Yes, but requires tuning
Cope with rotation of obstacle	Yes	No	No
Cope with translation of obstacle and end point	Yes	Yes	Yes

Table B.1: Overview of the conclusions

The simulation on the varying movement speed showed that the Generative Update method was not able to handle varying movement speeds or different movement speeds. This method generates a new GMM for every update based on the movement of the operator. If the operator movement is slightly different from the initial demonstrations, the reference trajectory adapts to shapes that are unusable as a reference trajectory. Direct Update can handle movements with varying movement speeds, but has also problems with movements of different speeds. This is due to the summed expectancy of the operator movement, which is based on the amount and expectancy of the data points in the operator movement.

The results of the simulation with the different environments is displayed in Figure B.5. All the methods can adapt the reference trajectory to the translated object and end point, but only Generative Update can adapt to the reference trajectory to the rotated obstacle.



(a) Generative Update (b) Generative Update

(c) Direct Update (d) Direct Update

(e) IGMM (f) IGMM

Figure B.5: Adapted trajectories for each method in different simulation environments.

B.2.2. Computational speed

The number of calculations that have to be done for each update is different for each method. To determine the relative difference in computation time, as well as the scaling of computation time with increasing data points, a series of simulations was performed.

The simulation environment of Figure B.4 was used for this simulation. The user movement was re-sampled to 10 data sets with a different amount of data points. The smallest set consisted of 50 data points and the biggest data set consisted of 500 data points. These data sets were used as inputs for the different adaptation methods that adapted the existing TP-GMM model. The simulation with each of the data sets was repeated 10 times in order to minimize the effect of potential influences that could affect the computation time. The results of the simulation are displayed in Figure B.6.

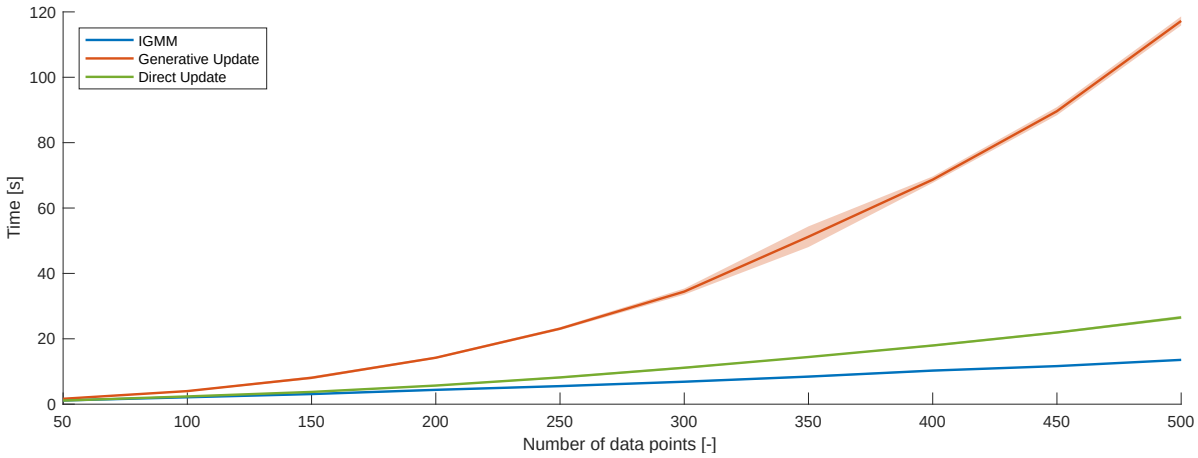


Figure B.6: Computation time in seconds for user movements with different amount of data points.

C

Munin Guide

This appendix will give a brief introduction to the Munin and might be useful as a starting point for future researchers.

C.1. Set-up

The Munin consists of a master and a slave that are connected by a real-time computer, which is labeled as the 'target computer'.Next to the target computer is the host computer, which has Windows 7 and several versions of MATLAB installed. The models that control the setup are built in Simulink and uploaded from the host computer to the target computer whenever the setup is going to be used.

C.1.1. Target computer

The target computer is a regular PC that has some extra PCI boards that have connections for controlling the servos and reading the sensors in the setup. The target computer boots from a USB stick, of which there are two versions. One of the sticks contains the real-time operating system that belongs to MATLAB2011b, while the other contains the real-time operating system that belongs to MATLAB2017b.

Figure C.1: The master (L) and slave (R) of the 'Munin' telemanipulator.

MATLAB2011b

The boot USB for this version is created using the two m-files. The first m-file `settings_xpcenv.m` is run to set all the variables for the target computer, while the second m-file `createbootdisk.m` is used to create the actual USB disk.

settings_xpcenv.m

```
%Set Environment Variables for XPC Target
setxpcenv('CDBootImageLocation','D:\haptics\Arnold\target\bootimage');
setxpcenv('BootFloppyLocation','O:');
setxpcenv('HostTargetComm','TcpIp');
setxpcenv('MaxModelSize','16MB');
setxpcenv('MulticoreSupport','off'); %NOTE: multicore support disabled, performance 2-3x
    better
setxpcenv('Name','MuninTarget2012');
setxpcenv('NonPentiumSupport','off');
setxpcenv('TargetRamSizeMB','2048');
setxpcenv('TcpIpTargetAddress','192.168.0.10');
setxpcenv('TargetMACAddress','F0:4D:A2:32:8B:A5');

setxpcenv('TargetBoot','BootFloppy');
env = getxpcenv;
disp([env.propname.', env.actpropval.']);
```

createbootdisk.m

```
disp('Create bootdisk program');

disp('Loaded settings');
settings_xpcenv;

createTrue = input('Are you sure you want to create a disk? (y/n)','s');
if strcmp(createTrue,'y')
    disp('Creating disk');
    xpcbootdisk;
else
    disp('No disk created');
end
```

MATLAB2017b

The command `slrtexplr` opens the Simulink Real Time Explorer, through which the boot USB can be created. The used settings for creating the USB stick are as follows:

Target Network Settings	
IP Adress	192.168.0.10
Subnet Mask	255.255.255.0
Port	22222
Gateway	255.255.255.255
Ethernet Device Settings	
Target Driver	Auto
Bus type	PCI
Target Settings	
USB Support	Yes
Graphics Mode	Yes
Boot mode	Removable Disk

Table C.1: Settings used for creating the MATLAB2017b USB boot disk

C.1.2. Host computer

The host computer controls the target computer via an ethernet cable. As mentioned before, there are several versions of MATLAB installed in the host computer. The model that is described in appendix D is built with MATLAB2011b and is therefore guaranteed to work with that MATLAB version. The model can also be used in MATLAB2017b, but it is not thoroughly tested. The only adjustments required in order for it to function in MATLAB 2017b is replacing '`obj.xpcObject = xpc();`' for '`obj.xpcObject = slrt();`' in line 133 of `Target_Model.m` and changing the 'memory allocation size'. If the memory allocation size isn't changed, the target computer will give an error stating: "allocation of logging memory failed". Therefore the buffer size is adjusted to the default 100000 in the Code Generation > Simulink Real-Time Options pane of the Configuration Parameters dialog box.

C.2. Step by step guide for using the model

The following steps apply for the used model and use with MATLAB2011b, since this is the MATLAB version that is used for the experiment. It might work the same for MATLAB2017b, but this has not been tested.

1. Run `startup.m`
2. Select `TPGMM_Final.mat` when the selection window opens. This is the file that contains the trained TP-GMM model.
3. Click 'Yes' when the 'Open Controller?' dialog opens. This will open the GUI that enables you to control the target computer.
4. Now press the 'Select' button in the GUI to select the Simulink model that is to be used. Choose `Munin_180529_Stijn_Final.mdl`, which is the model that will be described in appendix D.
5. Press the 'Upload' button to upload the selected model to the target computer.
6. Press the 'Refresh' button to update all the visuals in the GUI that represent the state of the target computer.
7. Start the electric amplifiers by pulling the red button upwards.

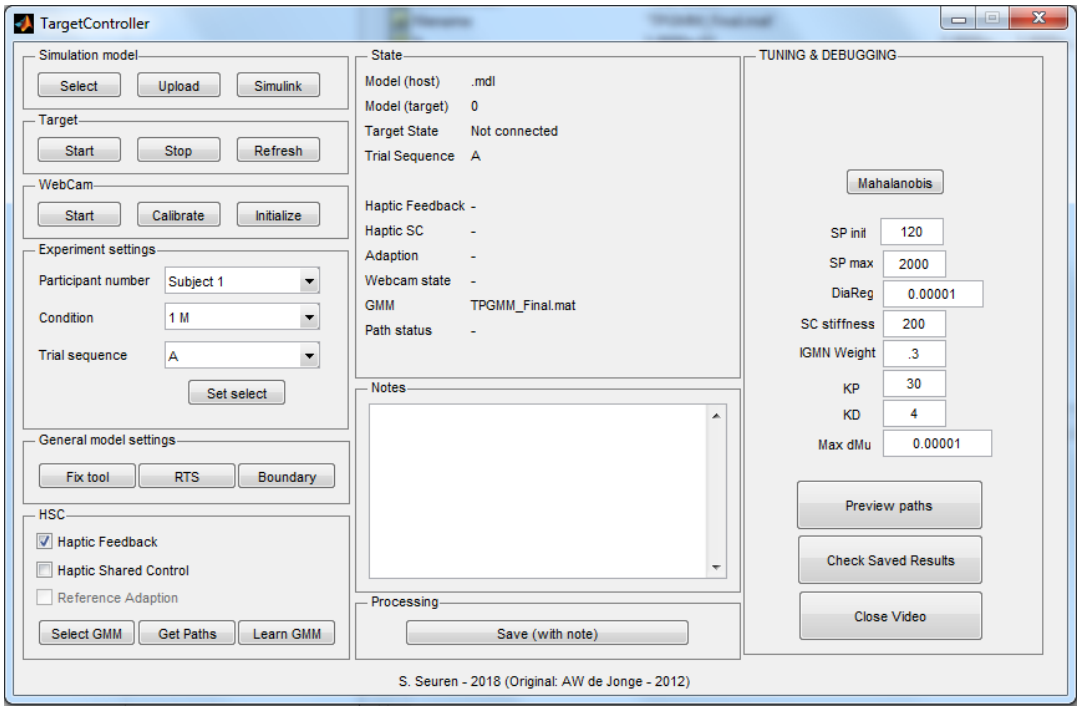


Figure C.2: The GUI for controlling the target computer

The model is now ready for use. The subject number, experiment condition and trial sequence can be changed to your preference. If the trial sequence is to be changed, the required sequence should be selected two times after each other. For unknown reasons the sequence only changes when it is selected for the second time.

When everything is set, the experiment can be started by following these steps:

1. Initialize the webcam by pressing the button in the GUI. This will open the slave-view window on the second screen that is attached to the host computer.
2. The Logitech webcam controller will open. Turn off the 'auto focus' function in the webcam options, otherwise the focus is adjusting continuously during the experiment.
3. Start the webcam by pressing the start button. This will initiate a loop that waits for the target computer to start the simulation.
4. Start the experiment by pressing the button that starts the model on the target computer.

Several buttons have been added to the lower left of the GUI for demo purposes. These buttons allow to turn on/off features of the model such as the haptic feedback, the HSC or the boundary crossing detection (which includes the vibration if the slave is outside the boundary).

D

Simulink model

This appendix will explain the structure in the Simulink model and the purpose of several blocks in the model. The nesting of the blocks can be seen in Figure D.1. The blocks highlighted with blue will be discussed in this chapter.

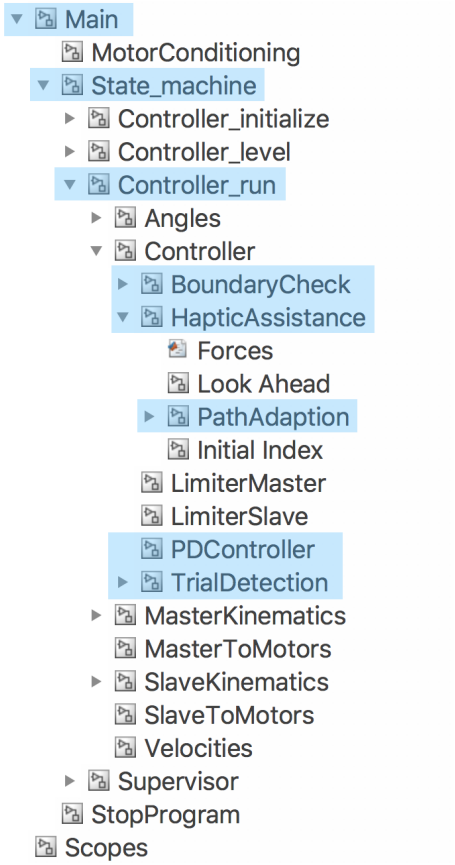


Figure D.1: Nesting of the Simulink model. The highlighted blocks will be discussed in this appendix.

D.1. Main

The top level of the model can be seen in Figure D.2. The inputs of the system are on the left side of the main block, while the outputs are on the right side of the main block. Inputs consist of the press buttons (used for calibration) and the rotary encoders. The outputs are the motor controls and the LED-lights on the setup.

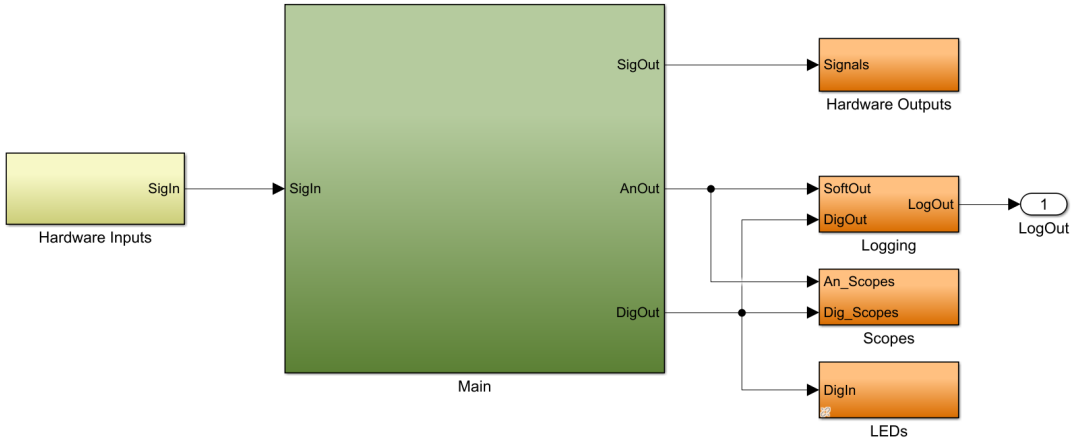


Figure D.2: Top level of the Simulink model

Opening 'Main' from the top level brings us to the 'State machine'-block that contains the core of the controller (Figure D.3, while the green 'MotorConditioning' block implements several safety limits to prevent the hardware from damaging itself.

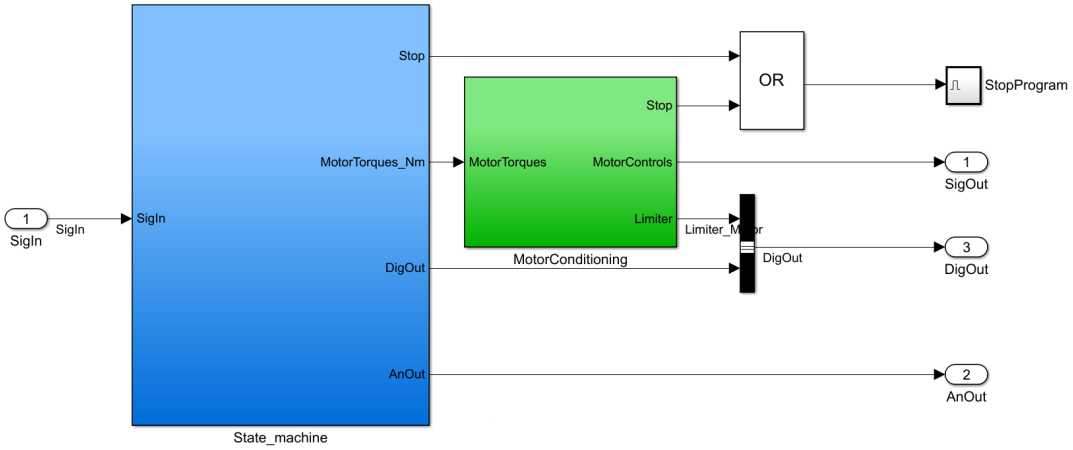


Figure D.3: Contents of the 'Main'-block.

D.2. State Machine

The state machine contains a 'supervisor' that determines when the system should switch between states. This system contains 3 different states: initializing, leveling and running.

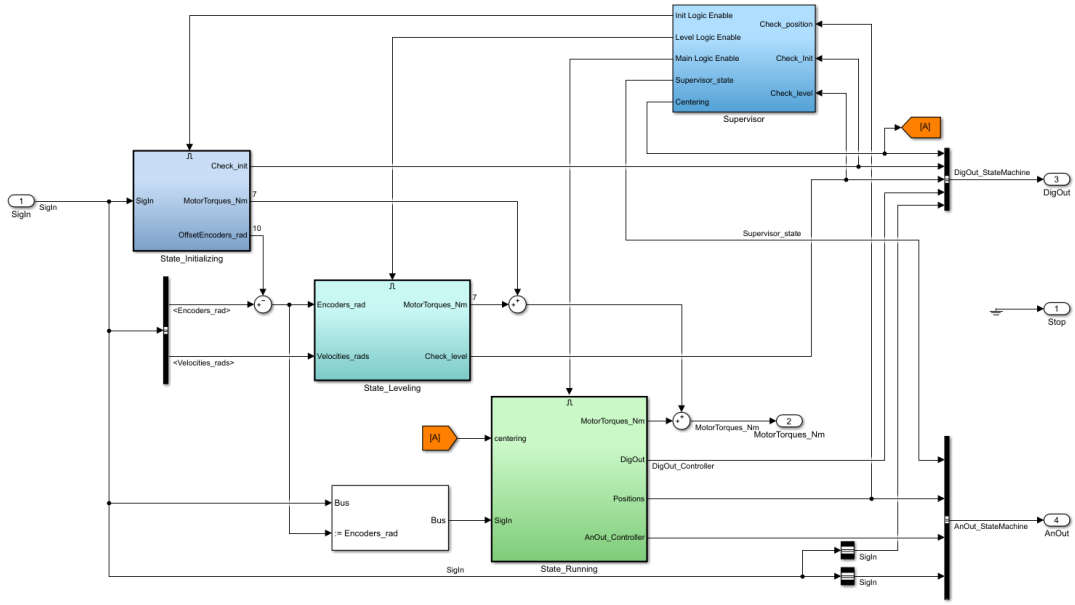


Figure D.4: The state machine

D.2.1. State: Initializing

The initializing state calibrates the encoders in the set-up. The master and slave are equipped with respectively 4 and 3 push buttons, one for each motor. This state actuates every motor until the beam attached to that motor touches a push button. By doing so, the offsets of the encoders are measured and with these offsets the system is calibrated.

D.2.2. State: Leveling

Once the system is calibrated, the supervisor will switch to the leveling state. This simply moves the master and slave to the center of their workspace. Once the master and slave are leveled, the supervisor switches to the running state.

D.2.3. State: Running

The running state contains the real controller. The controller is preceded by kinematic calculations that transform the measured rotations and torques to Cartesian locations and forces. The output of the controller is converted back to torques using the inverse kinematics.

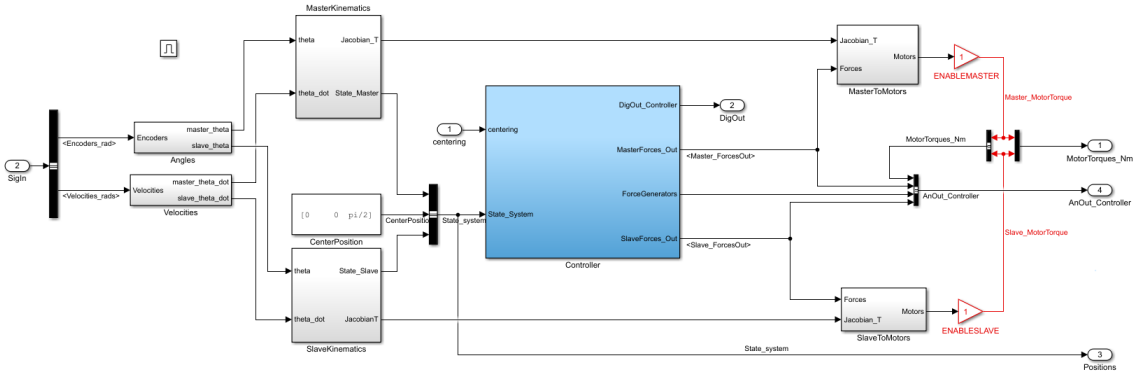


Figure D.5: Contents of the running state block, where angular inputs from the encoder are converted to Cartesian values. After the controller determined the force output, the forces are converted to torques using the inverse kinematics.

D.3. Controller

The controller is the heart of the model. There are 4 blocks that are capable of generating forces: the PD controller, the Haptic assistance, the boundary check and trial detection. Each of these blocks will be briefly discussed below.

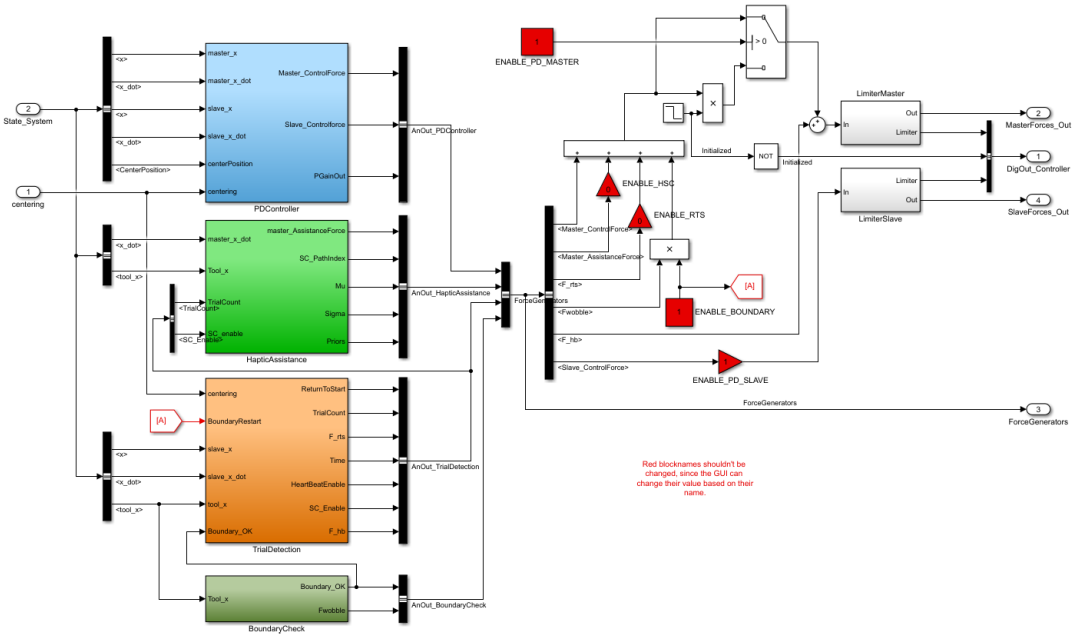


Figure D.6: These are the contents of the controller, which determines the output based on the input.

D.3.1. PD Controller

The PD controller minimizes the error between the master and the slave. A stiffness and damping are applied on the position and velocity error (K_p : 400, K_d : 0.02), which are then applied on both the slave and master. If the slave hits a physical object, an error will arise. This error is felt through the master device, but also results in a higher force exerted by the slave. This way the slave can interact with the environment while the operator receives force feedback.

D.3.2. Boundary Check

The boundary check block checks whether the slave is within the boundaries of the task environment. The block gives two outputs: a vibration force that acts on the master whenever the slave is outside the boundary and a Boolean 'BoundaryOK' that is true whenever the slave is within the boundaries.

D.3.3. Trial Detection

The trial detection block is very important and serves as a supervisor within the running state. Within the running state of the model, several states can be differentiated during the experiment:

- Slave in starting position, ready to start
- Trial is in progress and slave is moving towards finish
- Slave hit a boundary, returning to start
- Slave reached the end point and should return to start

These trials are detected using three different variables, knowing: 'HeartBeat', 'BoundaryOK' and 'TrialCount'. Heartbeat is true whenever the slave is within a 0.2cm radius of the starting position, BoundaryOK is true whenever the slave is within the boundaries and TrialCount adds 1 when the slave reaches the finish within a 0.2cm radius. Using derivatives of these signals, the current movement of the slave can be classified into one of the mentioned states. Also, the HSC is only turned on whenever the trial is in progress.

D.3.4. Haptic Assistance

The haptic assistance calculates the required guidance forces. These forces are generated by multiplying a stiffness (200N/m in this research) with the error between the reference trajectory and the estimated future location of the slave. The estimation of the future location is done in the 'Look Ahead' block, where also the look-ahead time can be adjusted. Based on the experiment condition, the reference trajectory can be adapted in the 'PathAdaption' block.

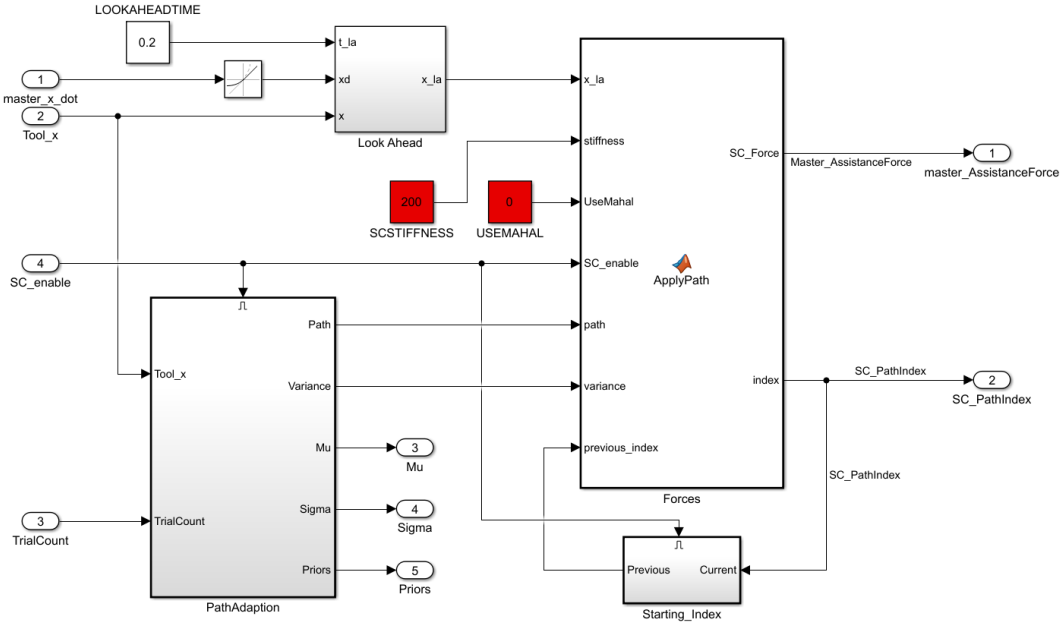


Figure D.7: The Haptic Assistance block

Path Adaption

The IGMM is implemented in this block. The reference trajectory is reset before each trial by the 'Initial Conditions' block. When the trial is started, the IGMM uses the slave position to adapt the reference trajectory online.

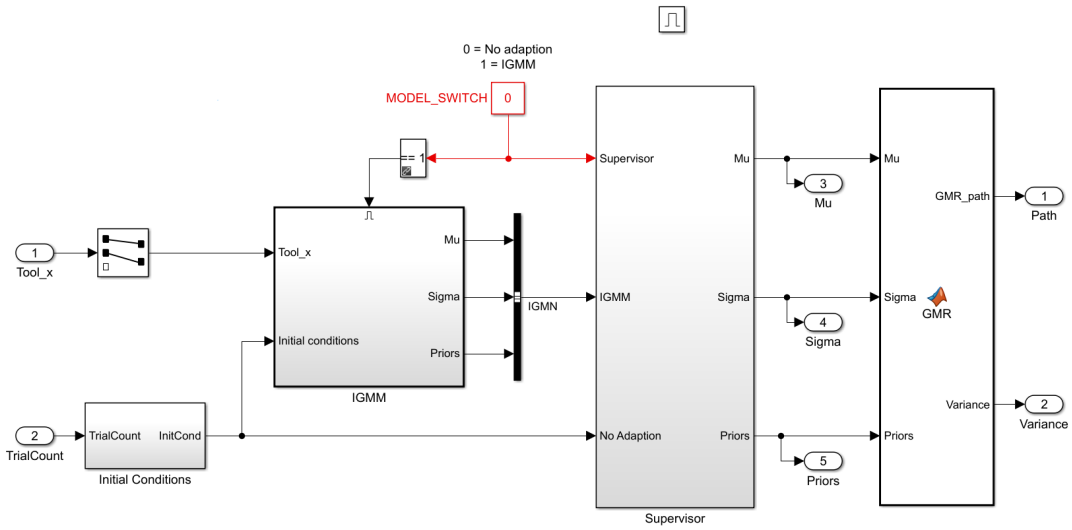


Figure D.8: Contents of the block that adapts the path in every update.

E

Plots of experiment data

E.1. Raw data plots

E.1.1. Raw data for manual

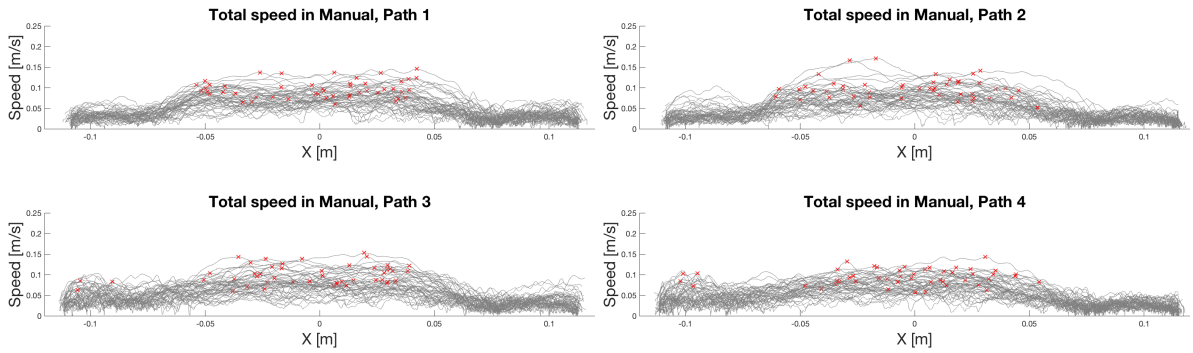


Figure E.1: Plots of the total speed during the trials of all the subjects. The red crosses show the maximum speed within one trial.

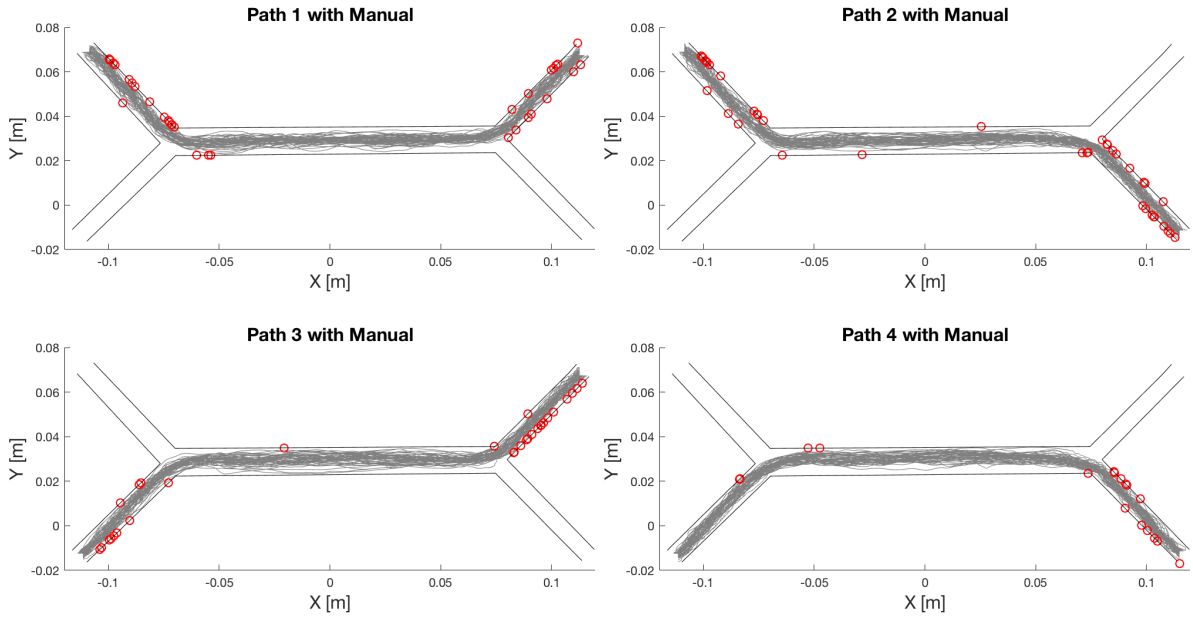


Figure E.2: The movement trajectories of all the subjects per path. The red circles show all the hit locations.

E.1.2. Raw data for cHSC

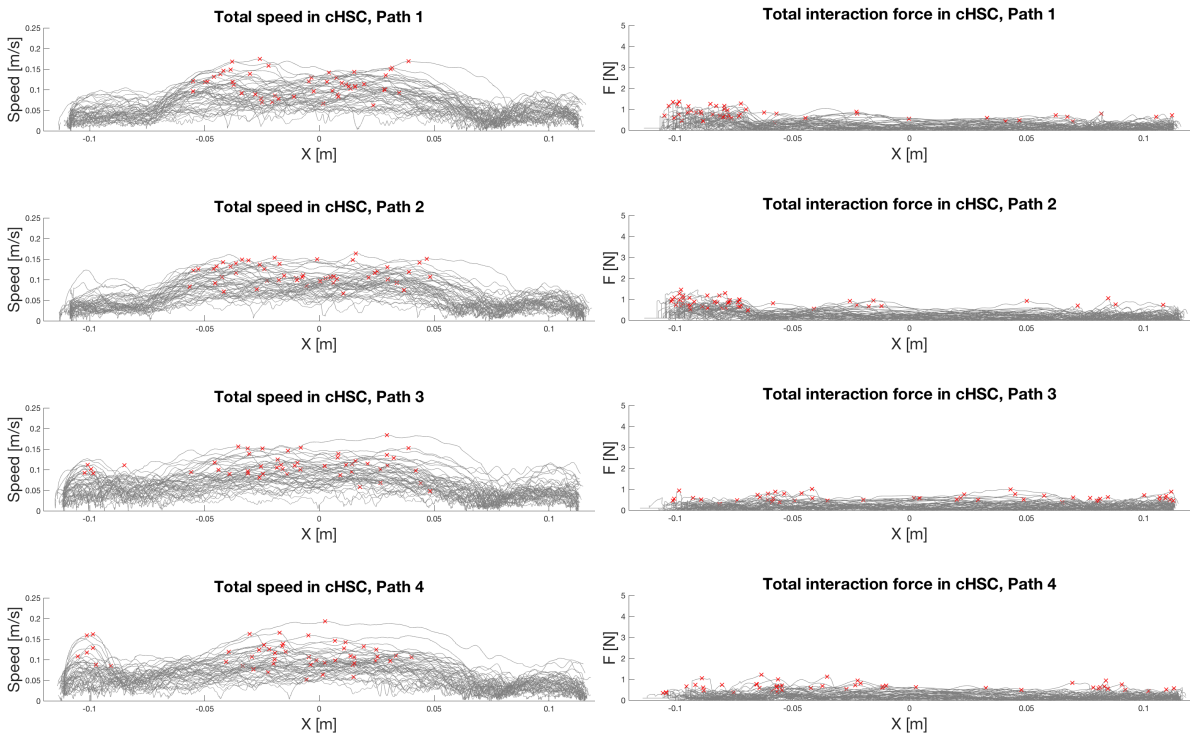


Figure E.3: Plots of the total speed and interaction force during the trials of all the subjects. The red crosses show the maximum speed and interaction force within one trial.

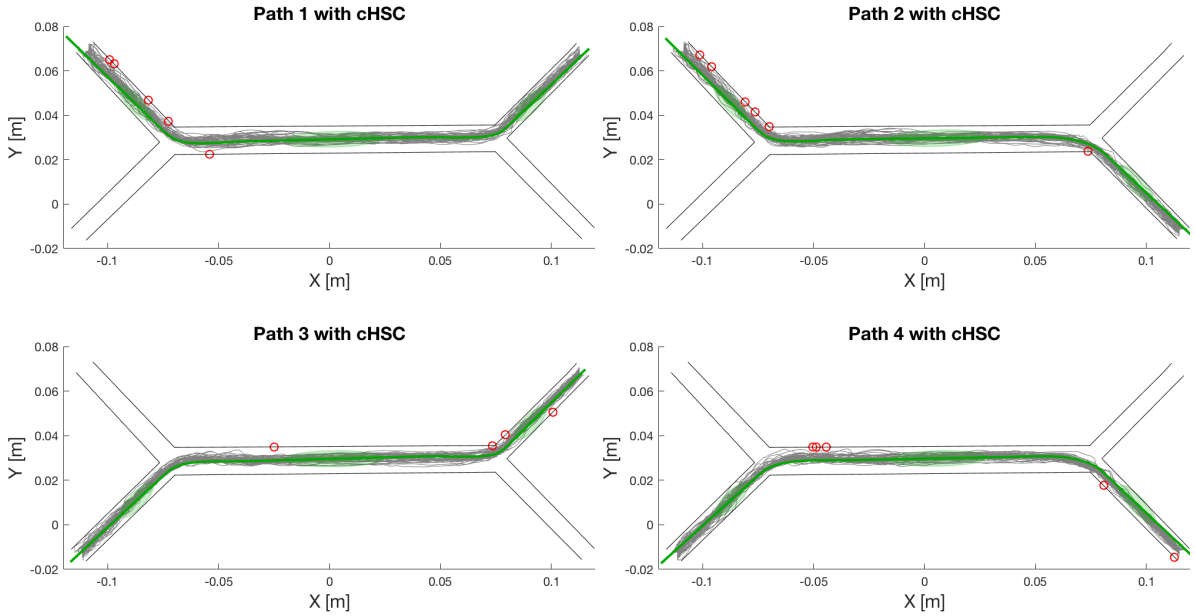


Figure E.4: The movement trajectories of all the subjects per path. The red circles show all the hit locations.

E.1.3. Raw data for wHSC

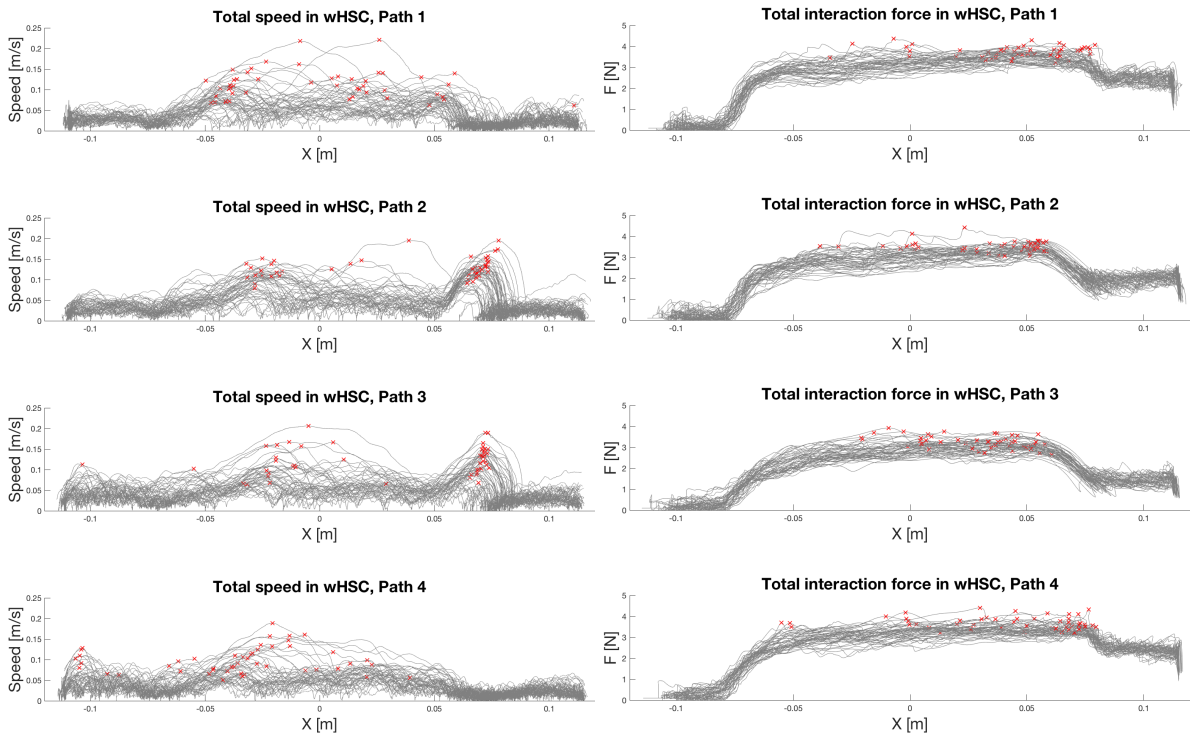


Figure E.5: Plots of the total speed and interaction force during the trials of all the subjects. The red crosses show the maximum speed and interaction force within one trial.

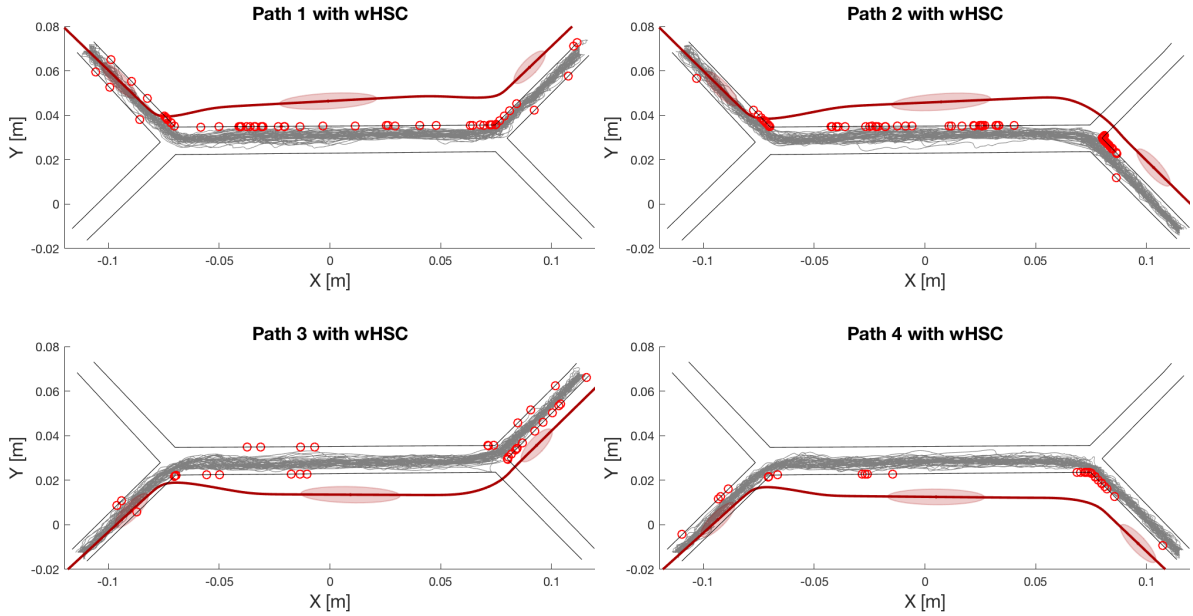


Figure E.6: The movement trajectories of all the subjects per path. The red circles show all the hit locations.

E.1.4. Raw data for aHSC

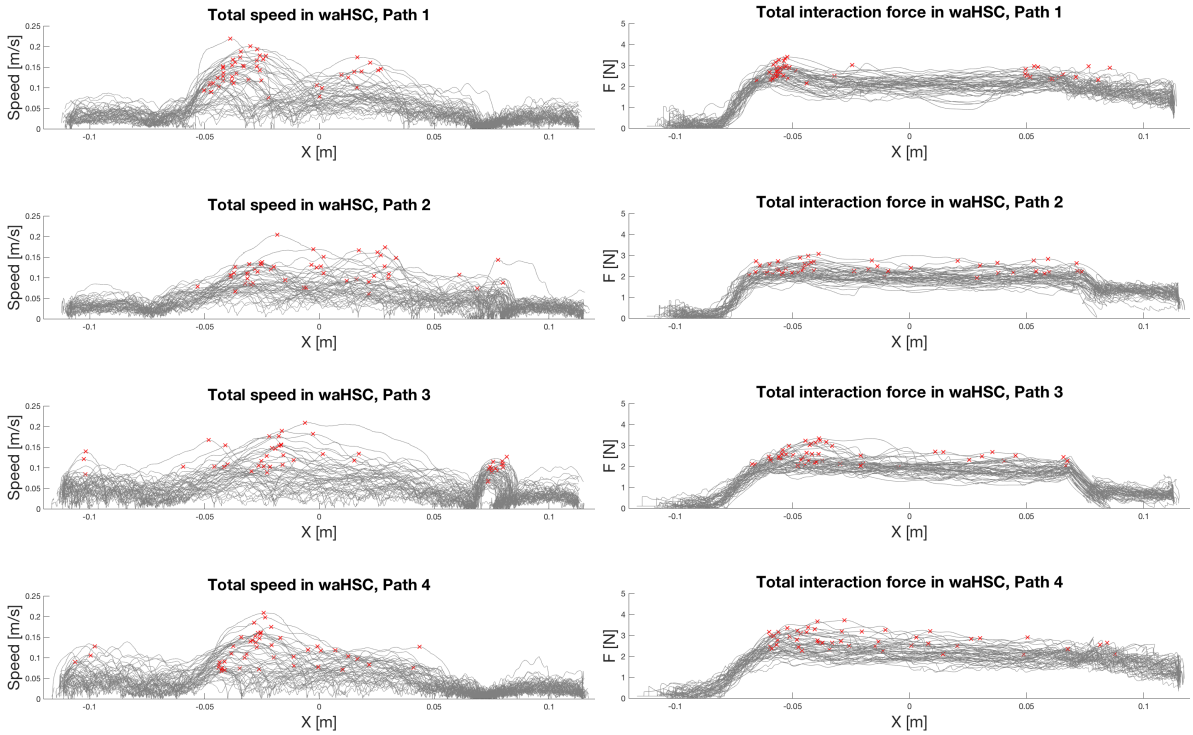


Figure E.7: Plots of the total speed and interaction force during the trials of all the subjects. The red crosses show the maximum speed and interaction force within one trial.

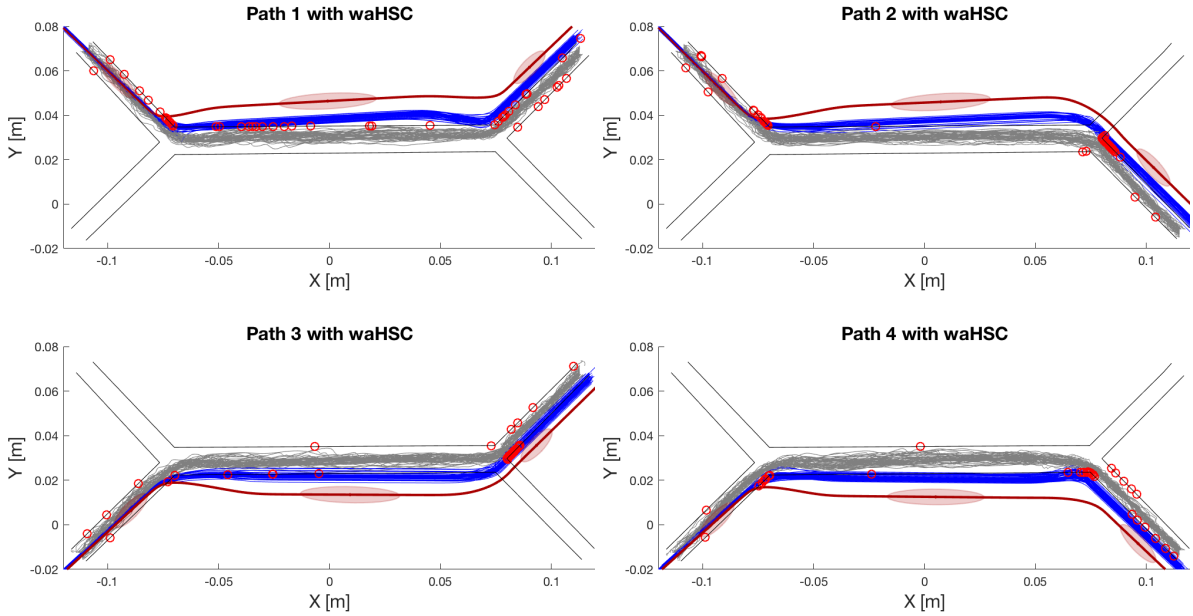


Figure E.8: The movement trajectories of all the subjects per path. The red circles show all the hit locations.

E.2. Metric plots

E.2.1. Per condition

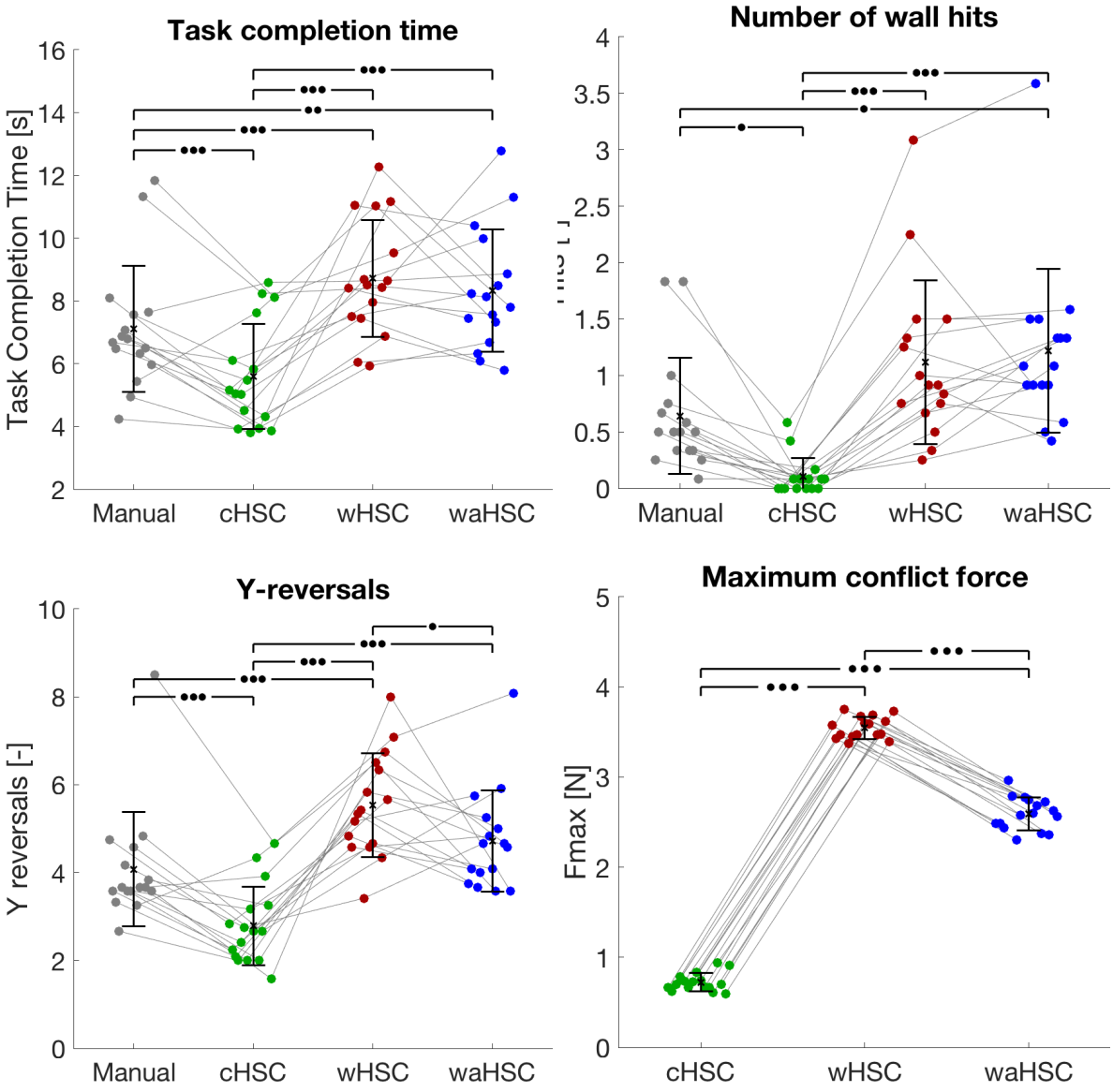


Figure E.9: Results for the different metrics per condition. The connected dots represent one subject and the significance is denoted by bullets (•), (••), (•••) that respectively represent p-values with $p \leq 0.05$, $p \leq 0.01$, $p \leq 0.001$ significance.

E.2.2. Per path

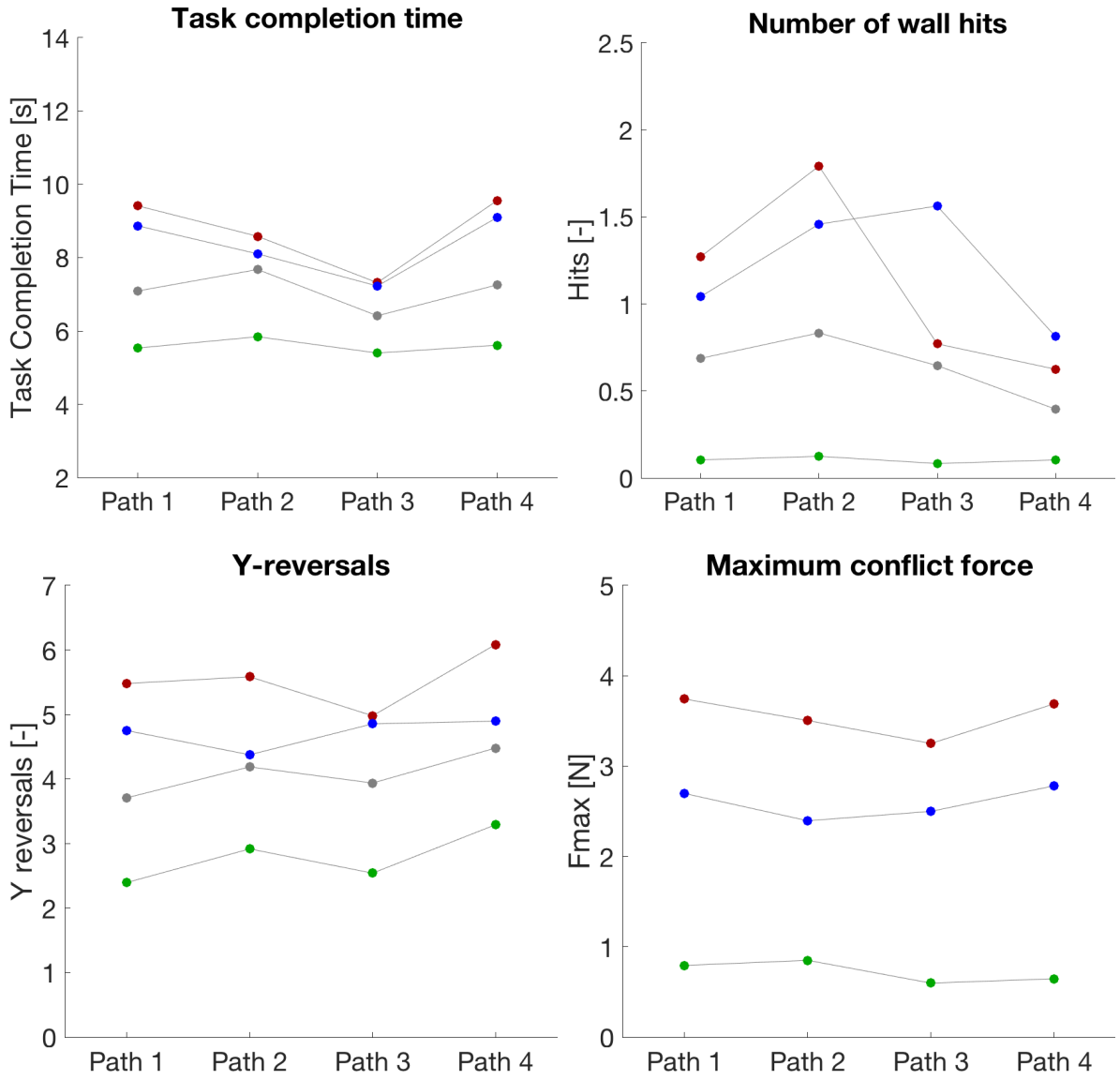


Figure E.10: Results for the different metrics per path. The red dots represent wHSC, blue represents waHSC, grey represents manual operation and green represents cHSC.

E.2.3. Per subject

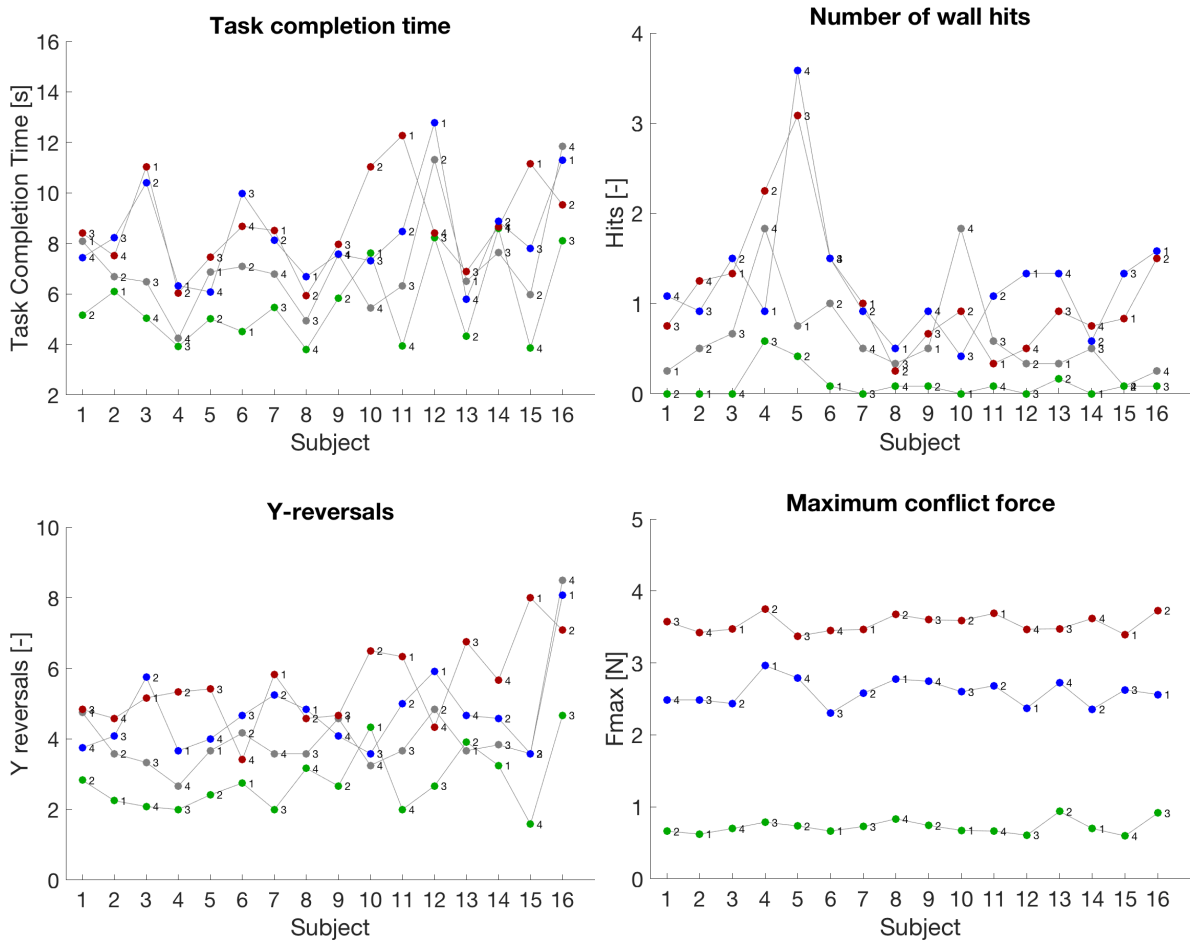


Figure E.11: Results for the different metrics per subject. The red dots represent wHSC, blue represents waHSC, grey represents manual operation and green represents cHSC. The number next to a colored dot shows the execution order of each condition per subject.

E.2.4. VanderLaan results

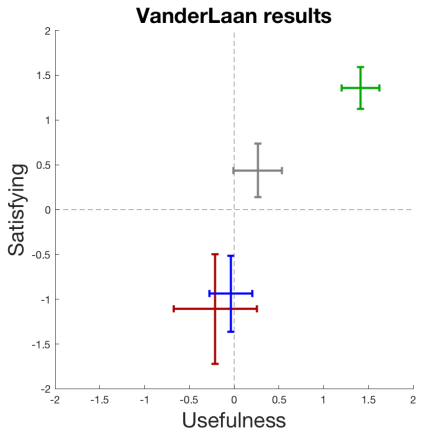


Figure E.12: The Van der Laan results for each condition. The red results represent wHSC, blue represents waHSC, grey represents manual operation and green represents cHSC.

E.3. Learning plots

E.3.1. Task completion time

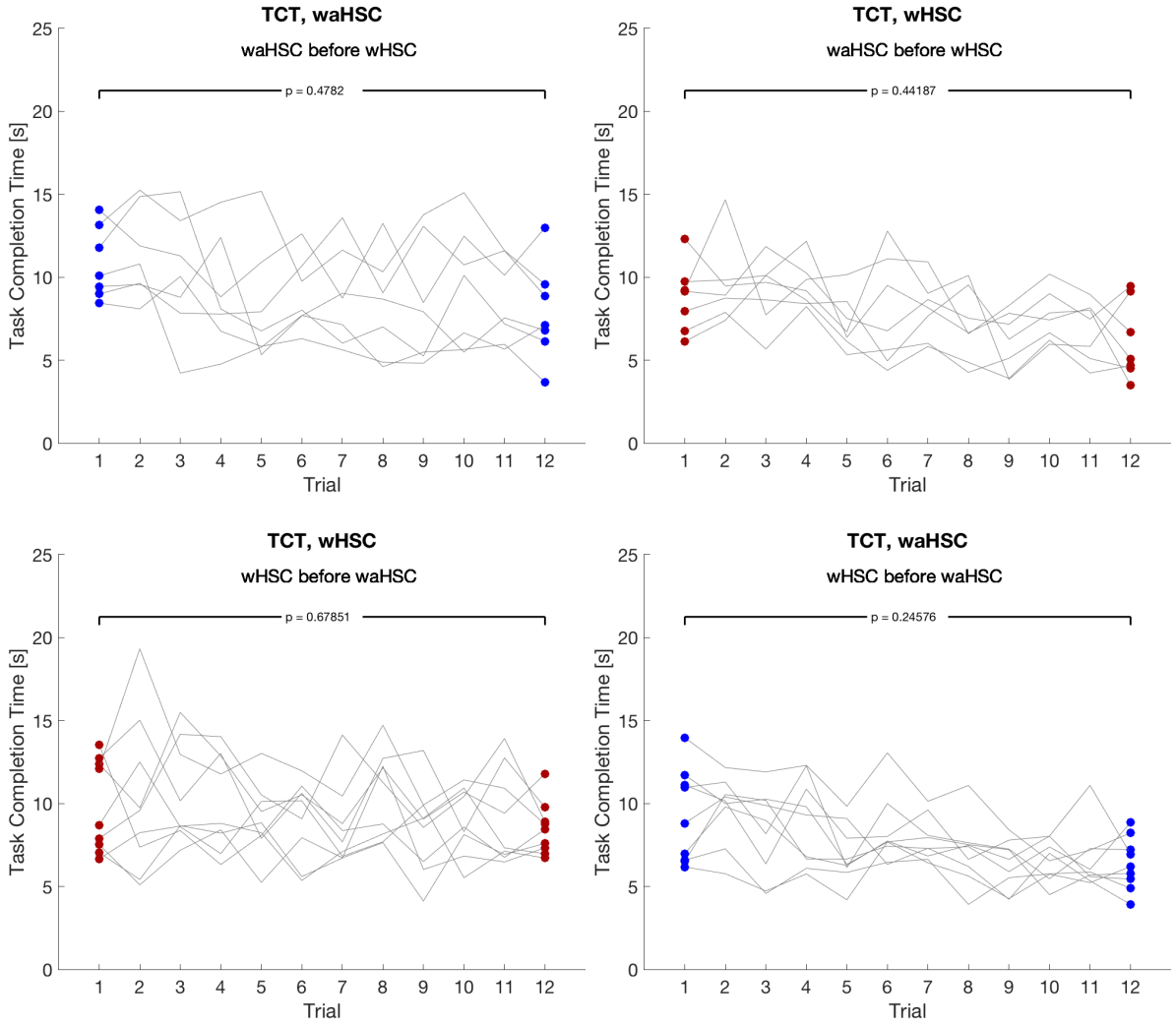


Figure E.13: These plots show the change of the results of all the subjects during the experiment and show possible strategy learning.

E.3.2. Number of wall hits

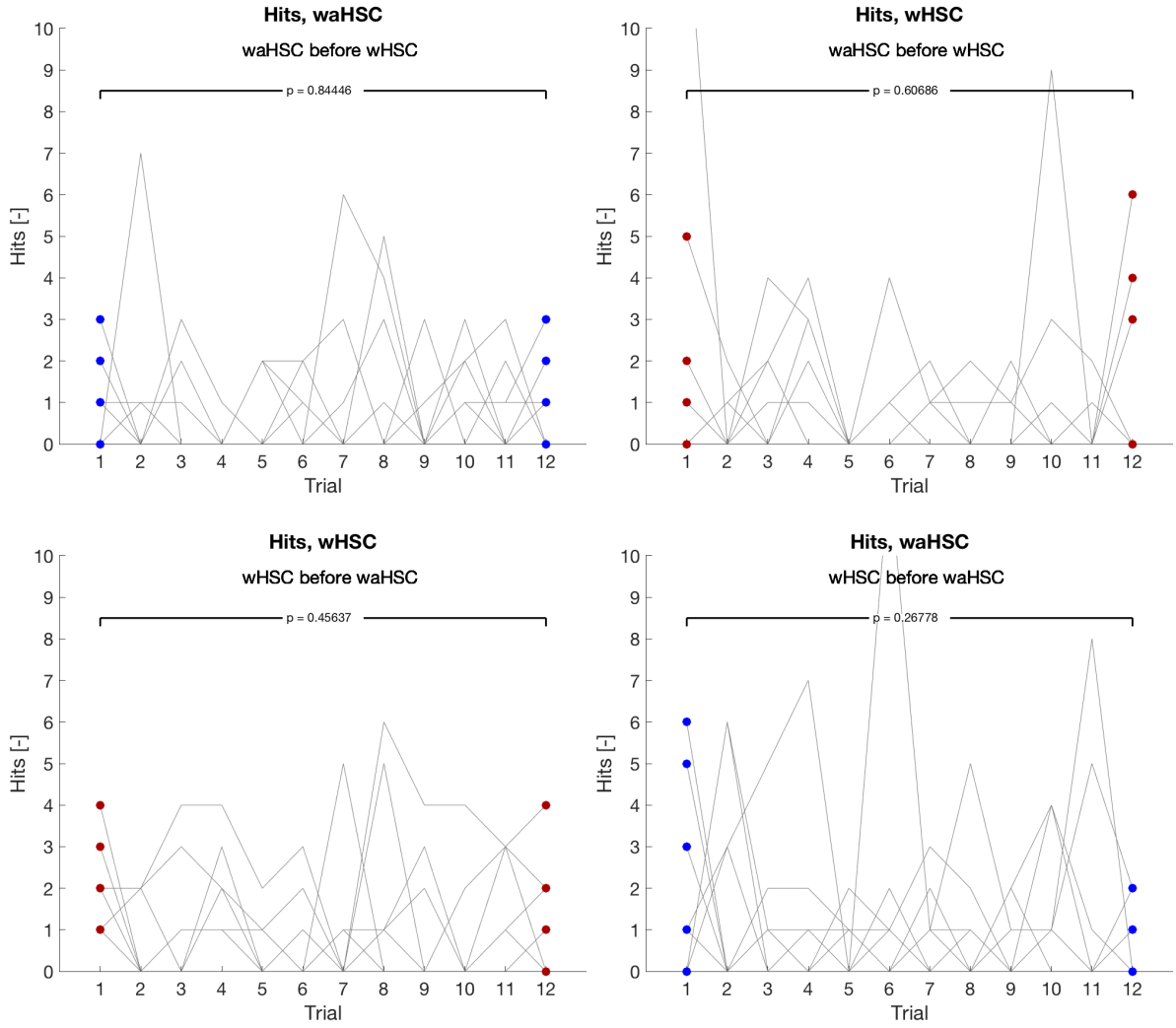


Figure E.14: These plots show the change of the results of all the subjects during the experiment and show possible strategy learning.

E.3.3. Y-reversals

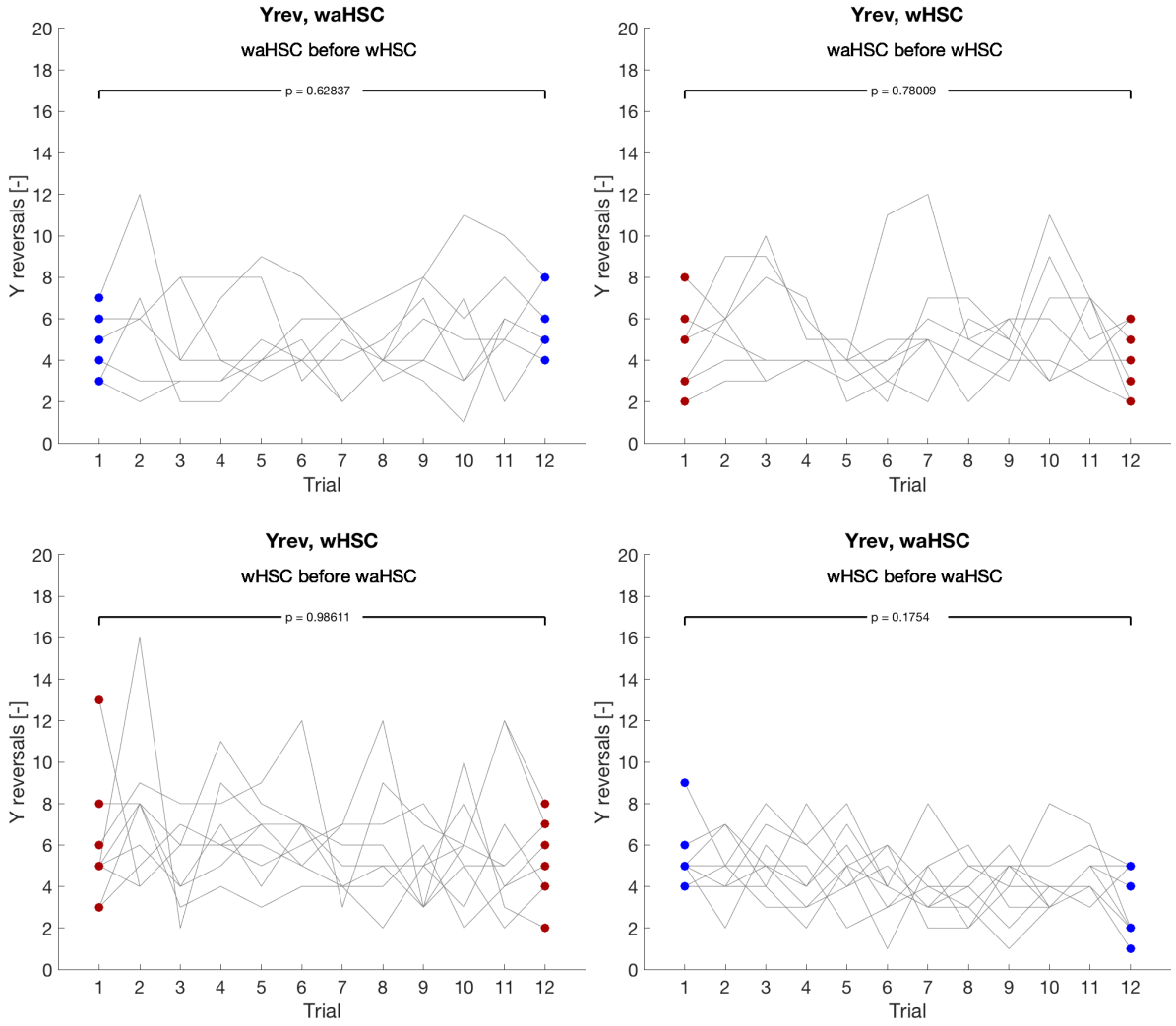


Figure E.15: These plots show the change of the results of all the subjects during the experiment and show possible strategy learning.

E.3.4. Maximum interaction force

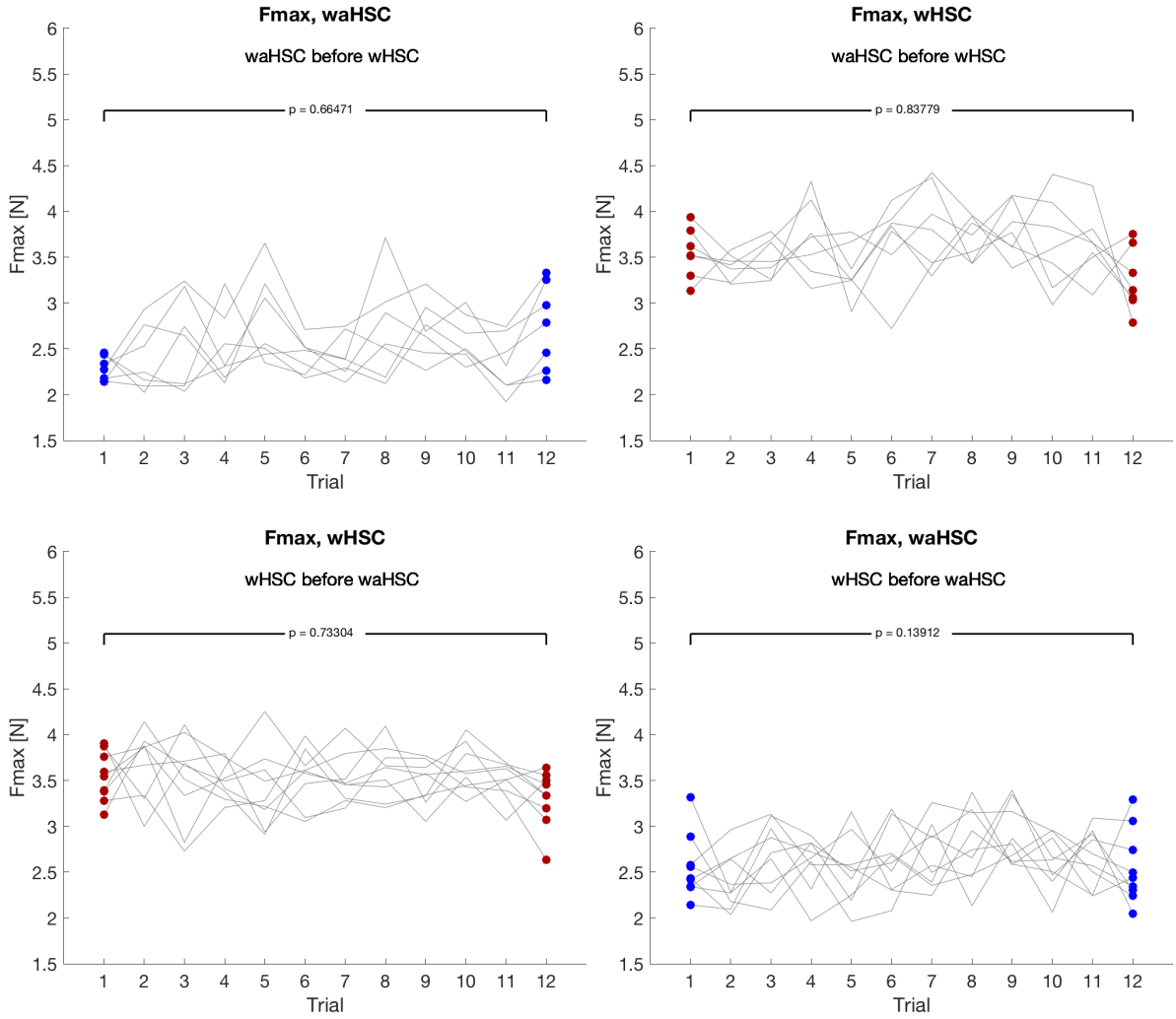


Figure E.16: These plots show the change of the results of all the subjects during the experiment and show possible strategy learning.

E.4. Order of execution differences

E.4.1. For each metric

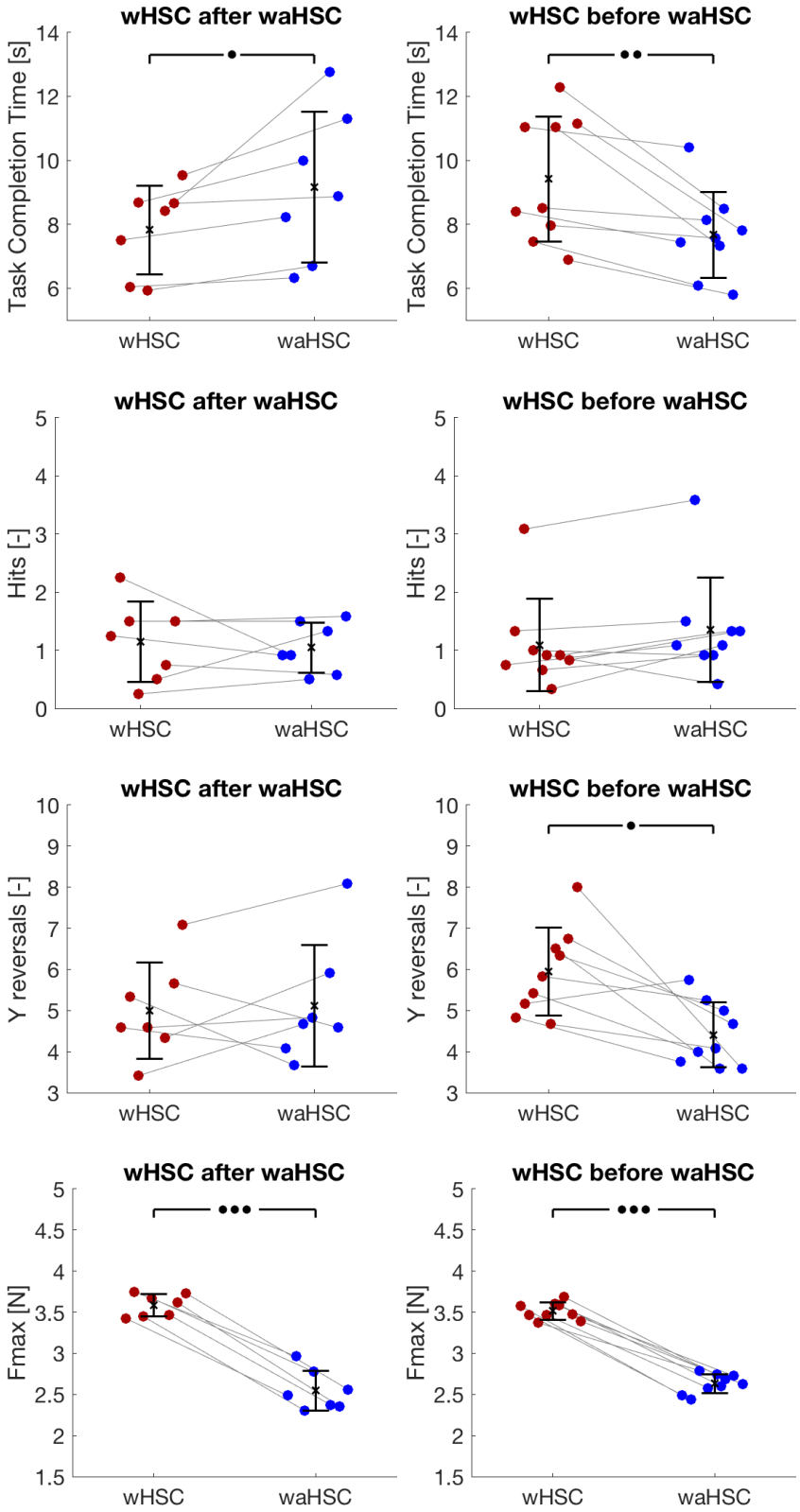


Figure E.17: Results for different order of execution of waHSC and wHSC. The bullets (•), (••), (• • •) respectively represent p-values with $p \leq 0.05$, $p \leq 0.01$, $p \leq 0.001$ significance.

E.4.2. For the VanderLaan results

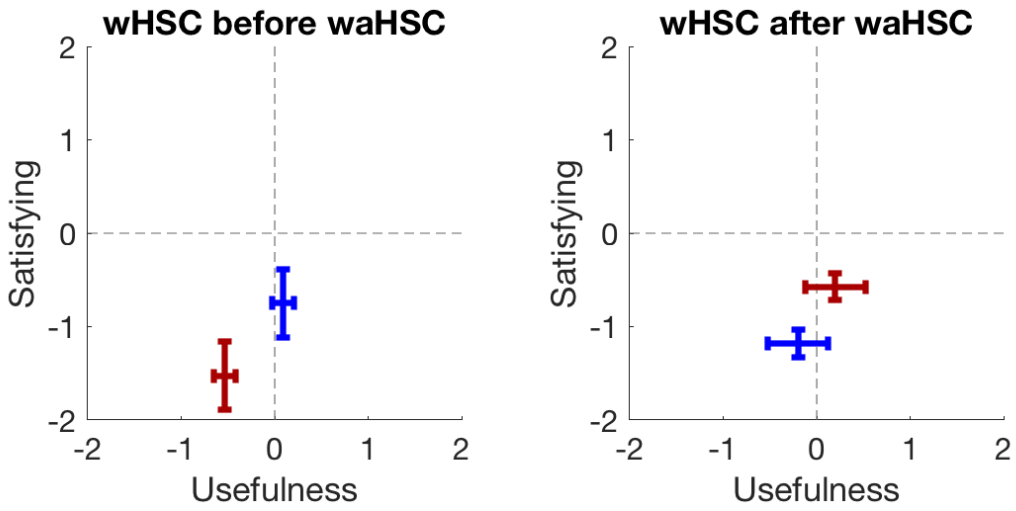


Figure E.18: Van der Laan results for different order of execution of waHSC and wHSC.

F

Ethics letter of approval

Date 18-06-2018
Contact person Ir. J.B.J. Groot Kormelink, secretary HREC
Telephone +31 152783260
E-mail j.b.j.grootkormelink@tudelft.nl

Human Research Ethics Committee
TU Delft
(http://hrec.tudelft.nl/)

Visiting address
Jaffalaan 5 (building 31)
2628 BX Delft

Postal address
P.O. Box 5015 2600 GA Delft
The Netherlands

Ethics Approval Application: Conflict based shared controller adaption using IGMN in teleoperation
Applicant: Boessenkool, Henri

Dear Henri Boessenkool,

It is a pleasure to inform you that your application mentioned above has been approved.

Good luck with your research!

Sincerely,

Prof. Dr. Sabine Roeser
Chair Human Research Ethics Committee TU Delft

Prof.dr. Sabine Roeser
TU Delft
Head of the Ethics and Philosophy of Technology Section
Department of Values, Technology, and Innovation
Faculty of Technology, Policy and Management
Jaffalaan 5
2628 BX Delft
The Netherlands
+31 (0) 15 2788779
S.Roeser@tudelft.nl
www.tbm.tudelft.nl/sroeser

G

Experiment Information sheet

Information sheet

Before agreeing to participate in this study it is important that the information in this document is carefully read and understood!
This document will describe the purpose, procedures, risks and possible discomforts of this experiment.

Purpose of the Research
The purpose of this research is to investigate the effect of different shared controllers in teleoperation on the performance, behavior and comfort of operators. The results will be statistically analyzed and published in a master thesis, as well as a possible publication.

The experiment setup
The haptic device ‘Munin’ is consists of a ‘Master side’, which is controlled by you, and a ‘Slave side’, which imitates the movements on the master side. The Master and Slave are connected via a computer that can help you perform a task using a shared controller.

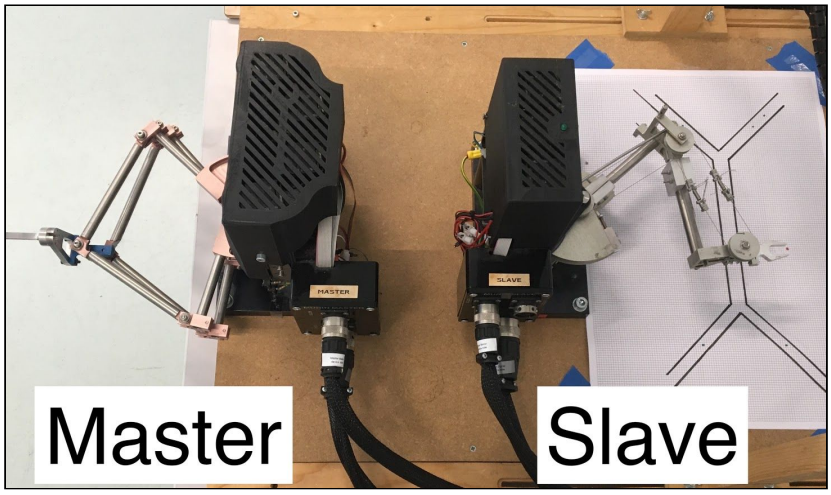


Figure 1: The “Munin”, a teleoperation experiment setup.

Procedure
After reading these instructions you will be asked to operate the Munin. First you will get the to familiarize yourself with controlling the Munin, after which the real experiment will begin. The experiment consists 4 sets of 12 trials and after each trial there will be a small questionnaire. The total required time for the experiment will be around 60 minutes.

Familiarization 12 trials	**Set 1** 12 trials 1 questionnaire	**Set 2** 12 trials 1 questionnaire	**Set 3** 12 trials 1 questionnaire	**Set 4** 12 trials 1 questionnaire

Task Instructions
Try to reach the indicated target as fast as possible without hitting the walls.

Using the setup
During the experiment you will make movements from the starting positions, located on the left side, to the end positions that are located on the right side of the screen. The webcam view will display red arrows to these start- and end positions. The webcam view will also display additional information, of which additional information can be found in the table below:

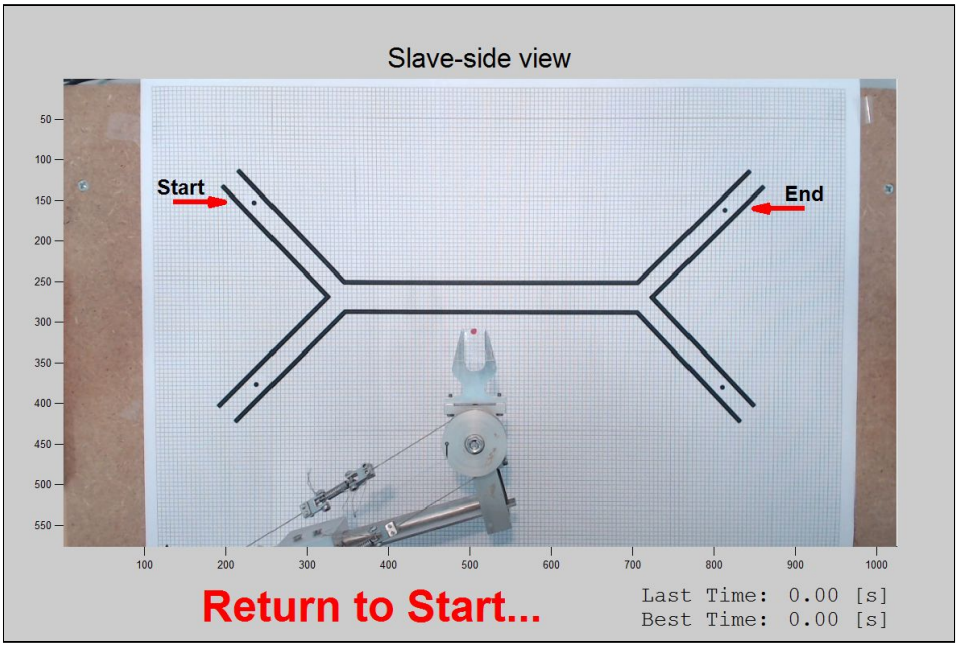


Figure 2: The Webcam view of the slave side

Started!	Try to move to the finish point as fast as possible
Start when ready	You will feel the 'heartbeat' vibration on the Master. You can start the trial when you are ready. If you started accidentally, you can move back to the start position for a restart.
Return to Start…	This will appear when you've reached the target. Move back to the next starting point.
Collision, restart	You just hit the wall, restart the trial.
Done!	You've completed the current set.

Benefits and risks of participating
You are asked to use an experiment setup that provides force feedback. During the experiment you can be exposed to forces up to 5 Newton.
There are no benefits or compensation for participating in this experiment.

Discomfort and withdrawal from the study
If you feel uncomfortable in any way or for any reason, you can press the red emergency button that is placed next to the Master device. At any time you have the right to withdraw from the experiment without mentioning any reason or consequences.

Confidentiality and privacy
This experiment will collect position and force data from the experiment setup as well as the answers from the questionnaires. All collected data will be:

- Anonymized, subjects are referred to by only a subject number.
- Kept confidential.
- Used for research purposes only.
- Stored in the TU Delft Haptics Lab repository for future learning and research. This repository is only accessible to researchers connected to the TU Delft Haptics Lab.

Questions
If you have any questions or complaints regarding the experiment or research you can contact Stijn Seuren, his contact details are listed below. Any questions about the nature of the different test-conditions can be asked at the end of the experiment.

Contact details

Researcher:
Stijn Seuren
s.a.seuren@student.tudelft.nl
+31 6 28521148

Supervisor:
Prof. Dr. Ir. D.A. Abbink
d.a.abbink@tudelft.nl

Supervisor:
Dr. Ir. M.J.A. Zeestraten
martijnzeestraten@gmail.com

H

Consent Form

Consent Form

Please tick the appropriate boxes	Yes	No
Taking part in the study		
I have read and understood the study information dated / / , or it has been read to me. I have been able to ask questions about the study and my questions have been answered to my satisfaction.	○	○
I consent voluntarily to be a participant in this study and understand that I can refuse to answer questions and I can withdraw from the study at any time, without having to give a reason.	○	○
I understand that taking part in the study involves operating an experiment setup with haptic feedback and filling out a questionnaire after each set of trials.	○	○
Risks associated with participating in the study		
I understand that taking part in the study involves the following risks: - Exposure to feedback forces up to 5 Newton	○	○
Use of the information in the study		
I understand that information I provide will be statistically analyzed and published in a master thesis, as well as a possible publication.	○	○
I understand that personal information collected about me that can identify me, such as age and gender, will not be shared beyond the study team.	○	○
Future use and reuse of the information by others		
I give permission for the anonymized questionnaire answers and experiment data that I provide to be archived in TU Delft Haptics Lab repository so it can be used for future research and learning. This repository is only accessible to researchers connected to the TU Delft Haptics Lab.	○	○

Signatures

_____	_____	_____
Name of participant	Signature	Date

I have accurately read out the information sheet to the potential participant and, to the best of my ability, ensured that the participant understands to what they are freely consenting.

_____	_____	_____
Researcher name	Signature	Date

Contact Details

Researcher	**Supervisor**	**Supervisor**
Stijn Seuren	Prof. Dr. Ir. D.A. Abbink	Dr. Ir. M.J.A. Zeestraten
s.a.seuren@student.tudelft.nl	d.a.abbink@tudelft.nl	martijnzeestraten@gmail.com
+31 6 28521148		

Subject questionnaire

Anonymous information

Could you please fill out the form below before the experiment? The subject number will be given to you by the researcher.

* Required

1. **What is your subject number?** *

2. **What is your gender?** *
   Mark only one oval.
   - ◯ Female
   - ◯ Male
   - ◯ Prefer not to say

3. **How old are you?**

4. **Are you left- or right handed?** *
   Mark only one oval.
   - ◯ Left handed
   - ◯ Right handed

5. **Have you ever operated a teleoperation device before?** *
   Mark only one oval.
   - ◯ Yes, but a different device
   - ◯ Yes, I have operated this device before
   - ◯ No

Powered by
Google Forms

Bibliography

[1] Sarcos, *Guardian GT,* (2017).

[2] S. Calinon, *A tutorial on task-parameterized movement learning and retrieval,* Intell. Serv. Robot. **9**, 1 (2016).

[3] S. Calinon and A. Billard, *Learning of Gestures by Imitation in a Humanoid Robot,* Imitation Soc. Learn. Robot. Humans Anim. Behav. Soc. Commun. Dimens. , 153 (2007).

[4] P. Engel and M. Heinen, *Incremental learning of multivariate Gaussian mixture models,* Adv. Artif. Intell. 2010 , 82 (2011).